DOCUMENT RESUME

ED 250 152 SE 045 108

TITLE Introduction to Number Systems, Boolean Algebra, Logic Circuits. Navy Electricity and Electronics Training Series. Module 13.
INSTITUTION Naval Education and Training Program Development Center, Pensacola, Fla.
REPORT NO NAVEDTRA-172-13-00-79
PUB DATE 79
NOTE 107p.
PUB TYPE Books (010) -- Guides - Classroom Use - Materials (For Learner) (051)

EDRS PRICE MF01/PC05 Plus Postage.
DESCRIPTORS Algebra; *College Mathematics; Higher Education; *Logic; *Mathematics; *Mathematics Instruction; *Number Systems; *Textbooks
IDENTIFIERS *Boolean Algebra; Logic Circuits; Navy

ABSTRACT

This textbook is one of a series of publications designed to provide information needed by Navy personnel whose duties require an elementary and general knowledge of the fundamental concepts of number systems, logic circuits, and Boolean algebra. Topic 1, Number Systems, describes the radix; the positional notation; the decimal, binary, octal, and hexadecimal number systems; and the conversion techniques needed to convert from one system to another. Topic 2, Boolean Algebra, includes rules, laws, mechanization, and simplification techniques. Topic 3, Logic Circuits, includes logic computation; logic polarity; and the six basic logic circuits of AND, OR, NOT, NAND, NOR, and Exclusive OR. A glossary of terms is included. (MNS)

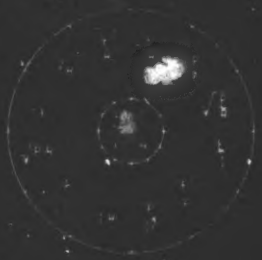
****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☒ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

ED250152

MODULE 13

NAVY ELECTRICITY AND ELECTRONICS TRAINING SERIES



INTRODUCTION TO

NUMBER SYSTEMS
BOOLEAN ALGEBRA
LOGIC CIRCUITS

NAVAL EDUCATION AND TRAINING COMMAND

NAVEDTRA 172 13 00 79

SE045108

Although the words "he", "him", and "his", are used sparingly in this manual to enhance communication, they are not intended to be gender driven nor to affront or discriminate against anyone reading *Module 13, Introduction to Number Systems, Boolean Algebra, and Logic Circuits*, NAVEDTRA 172-13-00-79.

PREFACE

Introduction to Number Systems, Boolean Algebra, and Logic Circuits is part of the Navy Electricity and Electronics Training Series (NEETS). This module and its associated Officer-Enlisted Correspondence Course (OCC-ECC) provides the information needed by personnel of the U. S. Navy and Naval Reserve whose duties require an elementary and general knowledge of the fundamental concepts of number systems, logic circuits, and Boolean algebra.

Topic 1, Number Systems, describes the radix, the positional notation, the decimal, binary, octal, and hexadecimal number systems, and the conversion techniques needed to convert from one system to another. Topic 2, Boolean Algebra, includes rules, laws, mechanization and simplification techniques. Topic 3, Logic Circuits, includes logic computation, logic polarity, and the six basic logic circuits—AND, OR, NOT, NAND, NOR, and Exclusive OR.

Module 13 has been written by and with the advice of senior Navy technicians. The information contained in this module is designed to give you (the student) small amounts of information at a time so that you can easily digest it before going on to more complex material.

Questions are distributed throughout the text. These questions are designed to assist you in determining if you understood the preceding information.

If there is any difficulty in phrasing an answer, you should restudy the applicable paragraph(s). Answers to the questions are at the top of the next even-numbered page.

Before attempting this course, you should already have an understanding of basic electronics, including solid-state theory and mathematics through high school algebra. A review of applicable chapters in *Mathematics, Volume 1*, NAVEDTRA 10069 (Series), and *Mathematics, Volume 3*, NAVEDTRA 10073 (Series) will be of great assistance to you in completing this module.

This Module and associated OCC-ECC were prepared and will be administered by the Naval Education and Training Program Development Center, Pensacola, Florida, for the Chief of Naval Education and Training.

Your suggestions and comments on the *NEETS* are invited. Address them to NETPDC, Code PD, Pensacola, FL 32509.

1979

Stock Ordering No.
0507-LP-172-1300

Published by
NAVAL EDUCATION AND TRAINING PROGRAM
DEVELOPMENT CENTER

UNITED STATES
GOVERNMENT PRINTING OFFICE
WASHINGTON, D.C.: 1979

THE UNITED STATES NAVY

GUARDIAN OF OUR COUNTRY

The United States Navy is responsible for maintaining control of the sea and is a ready force on watch at home and overseas, capable of strong action to preserve the peace or of instant offensive action to win in war.

It is upon the maintenance of this control that our country's glorious future depends; the United States Navy exists to make it so.

WE SERVE WITH HONOR

Tradition, valor, and victory are the Navy's heritage from the past. To these may be added dedication, discipline, and vigilance as the watchwords of the present and the future.

At home or on distant stations we serve with pride, confident in the respect of our country, our shipmates, and our families.

Our responsibilities sober us; our adversities strengthen us.

Service to God and Country is our special privilege. We serve with honor.

THE FUTURE OF THE NAVY

The Navy will always employ new weapons, new techniques, and greater power to protect and defend the United States on the sea, under the sea, and in the air.

Now and in the future, control of the sea gives the United States her greatest advantage for the maintenance of peace and for victory in war.

Mobility, surprise, dispersal, and offensive power are the keynotes of the new Navy. The roots of the Navy lie in a strong belief in the future, in continued dedication to our tasks, and in reflection on our heritage from the past.

Never have our opportunities and our responsibilities been greater.

CONTENTS

TOPIC	Page
1. Number Systems	1
2. Boolean Algebra	20
3. Logic Circuits	71
APPENDIX	
I. Glossary of Terms	91
INDEX	98

TOPIC 1

NUMBER SYSTEMS

You are about to enter the fascinating world of computers. Many advances in computer technology have been made in the last two decades. Today it is difficult to find a field of human endeavor that has not been affected in some way by computer technology. When you complete this module, you will have an understanding of the basic principles underlying all computers.

Why are you going to study number systems? After all, you know how to add and subtract, and probably a lot more about how to use numbers than you realize. The problem is, you are used to one particular set of numbers: the decimal system. However, numbers are written in many different ways.

Since people and computers do not speak the same language, methods of translating information into a form which is understandable and usable to both are necessary. Humans generally speak in words and decimal numbers, while computers only understand coded electronic pulses which represent digital information.

This first topic will describe numbering systems in general; and binary, octal, and hexadecimal number systems specifically. Methods for converting numbers in the binary, octal, and hexadecimal systems to equivalent numbers in the decimal system (and vice versa) will also be described.

COMPUTERS

Computers have made possible military, scientific, and commercial advances that were considered to be impossible only a few years ago. The mathematics involved in orbiting a satellite about the sun, for example, would occupy several teams of mathematicians for a lifetime. Now, with the aid of electronic digital computers, the conquest of space has become a reality.

Computers are now employed wherever repetitious calculations or the processing of huge amounts of data are necessary. The greatest applications are found in the military, scientific, and commercial fields. They are used in many varied projects, ranging from mail sorting, through engineering design, to the identification and destruction of enemy targets. The advantages of digital computers include speed, accuracy, and man-power savings. Frequently computers are able to take over routine jobs, releasing men for more important work, work that can not be handled by a computer.

HISTORICAL BACKGROUND

Ever since people discovered that it was necessary to count objects, they have been looking for easier ways to do it. Contrary to popular belief, digital computers are not a new idea. The abacus is a manually operated digital computer used in ancient civilizations, and utilized to this day in the Orient. For those who find this humorous, it is interesting to note that in a contest between a modern desk calculator and an abacus, the abacus won.

The first adding machine was invented by Blaise Pascal (French) in 1642. Twenty years later an Englishman, Sir Samuel Morland, developed a more compact device which could multiply, add, and subtract. In 1682, Wilhelm Liebnitz (German), perfected a machine which could perform all the basic operations (add, subtract, divide, multiply), as well as extract the square root. Liebnitz's principles are still in use today in our modern electronic digital computers.

As early as 1919, electronics entered the scene. An article by W. H. Eccles and F. W. Jordan described an electronic "trigger circuit" that could be used for automatic counting. But the ECCLES-JORDAN multivibrator was a little ahead of its time.

Today every digital computer employs these circuits, known as flip-flops, to store information, perform arithmetic operations, and control the timing sequences within the computer.

Under the pressure of military needs in World War II the science of electronics data processing made giant strides forward. In 1944, Harvard University developed a computing system known as the Automatic Sequence Controlled Calculator. After the initial design and construction, several improved models were built.

Meanwhile, at the University of Pennsylvania, a second system was being developed. This system, completed in 1946, was named ENIAC (Electronic Numerical Integrator and Computer). ENIAC employed 18,000 vacuum tubes in its circuitry, but in spite of this worked quite successfully. The first problem assigned to ENIAC was a calculation in nuclear physics which would have taken 100 man-years to solve by conventional methods. ENIAC solved the problem in two weeks, only two hours of which was actually spent on calculation. The remainder of the time was spent checking the results and operational details. All modern computers are based on the principles used in these early developments.

In 1950, UNIVAC was developed. This machine was regarded as the most successful electronic data processor of its time. One of the most outstanding features of the UNIVAC was that it checked its own results in each step of a problem, thus eliminating the need to run the problems more than once to insure accuracy.

During the first outbreak of publicity about computers (especially when the UNIVAC predicted the outcome of the 1952 presidential election), the term "giant brain" caused a great deal of confusion and uneasiness in people. Many people assumed that science had created a thinking device superior to the human mind. Today most people know better. By human standards the giant brain is nothing more than a talented idiot that is wholly dependent upon human instructions to perform even the simplest job. A computer is only a machine and definitely can not think for itself.

The field of digital computers is still in the developmental stages. New types of circuitry and new ways of accomplishing things are continually developing at a rapid rate.

In the military field the accomplishments of digital computers are many and varied. One example is the guided missile weapons systems. Most of the navigation of the nuclear powered submarine is done by a digital computer and this system is highly successful.

Another military application of digital computers is found in our supply system. Computers are used to account for supplies in such a manner that the Navy Supply System always knows how much of an item is on hand, where it is, and when more of that item should be procured.

ONE-TO-ONE COUNTING

The South African Hottentot probably has the simplest aritnmetic system in the world: "One, two, plenty" is as far as it goes. Most cultures have a somewhat more sophisticated method of counting: the one-to-one system. A primitive tribesman, for example, might count time by dropping a stone into a pot for each passing day. If the pot holds seven stones, a full pot marks one week. Observe that one stone represents one day, a one-to-one relationship.

The meter of the taxi driver in ancient times provides another example. The driver carried a box of pebbles with him. It was constructed so that it would drop a stone into a plate every few minutes. When the ride was over the driver billed his fare by showing him the number of pebbles in the plate. Again, one pebble stood for one unit of time, a one-to-one relationship.

Sticks, marks on the earth, and many other objects were used as tools for this one-to-one counting method. One tool, however, was more convenient than the others. Discovered early in history, this counting tool is so obvious that everyone has used it. You have it available at all

times. This tool is your ten fingers. This is believed to be the origin of the decimal system.

DECIMAL, BINARY, OCTAL, AND HEXADECIMAL NUMBER SYSTEMS

Before studying any particular number system, it would be best for you to examine number systems in general, to discover how they work. All modern number systems are built from the following components: the UNIT, the NUMBER, and the RADIX.

1. UNIT—The unit is a single object.

2. NUMBER—The number is an arbitrary symbol or group of symbols representing a unit or sum of units. Both 5 and V are numbers and represent a sum of five units.

3. RADIX—The radix is the base of a positional number system. It is equal to the number of symbols needed to count from zero to the radix minus one or (R-1) of the number system. The largest number that can be written in any position is the radix minus 1 or (R-1). In the decimal system the radix is 10; that is, 10 symbols are needed to count from zero to R-1 or 0 through 9. The octal system (base 8) requires eight symbols (0 through 7), and the binary system (base 2) requires two symbols, (0 and 1).

Decimal System

Since the decimal system (also known as the Hindu-Arabic system) uses ten symbols or digits (See TABLE 1-1.), it has a radix or base of 10. This system is thought to have evolved and found common usage as a result of our having ten fingers (digits).

Because the decimal system is used almost universally, basic arithmetic performed by a person in one country is easily understood by a person in another country. In other words, the decimal system serves as a sort of universal language. Because of its common usage and because of its relationship with other number systems, this system will serve as a basis for discussion of the other number systems.

Table 1-1.—Comparison of Commonly Used Number Systems

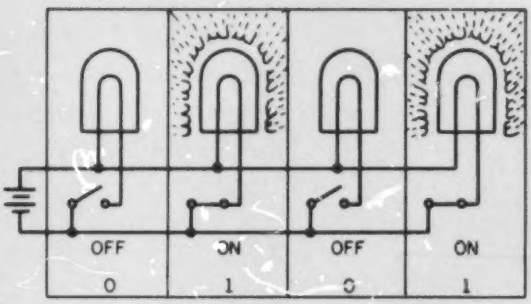
BINARY	OCTAL	DECIMAL	HEXADECIMAL
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F
10000	20	16	10
10001	21	17	11
10010	22	18	12
10011	23	19	13
10100	24	20	14
10101	25	21	15
10110	26	22	16
10111	27	23	17
11000	30	24	18
11001	31	25	19
11010	32	26	1A
11011	33	27	1B
11100	34	28	1C
11101	35	29	1D
11110	36	30	1E
11111	37	31	1F
100000	40	32	20
100001	41	33	21
100010	42	34	22
↓	↓	↓	↓

Binary System

The simplest possible number system is based on powers of two and is known as the binary system. Table 1-1 illustrates the relation

between the binary system, the decimal system, and other commonly used systems.

By a convenient coincidence, the two binary conditions (1 and 0) can easily be represented by many electrical/electronic devices. For example, the binary 1 state may be indicated when the device is active and the 0 state may be indicated when the device is nonactive.



Examine the inserted figure above. This figure illustrates a very simple binary counting device. Notice that binary 1 is indicated by a lighted lamp, and binary 0 is indicated by an unlighted lamp. The reverse will work equally well; i.e., the unlighted state of the lamp can be used to represent a binary 1 condition and the lighted state can represent the binary 0 condition. Both methods are used in digital computer applications. There are numerous other devices used to represent binary conditions. These include switches, relays, diodes, transistors and integrated circuits (ICs).

Octal System

The octal system has eight distinct characters (TABLE 1-1), hence its radix is 8. The octal system is quite useful as an accessory to the binary system because 8 is an integral power of two (2). That is, $8_{10} = 2^3_{10}$. One octal digit has a value equivalent to that of a group of three binary digits and vice versa, as indicated below.

Octal to Binary			Binary to Octal		
2	2	5_8	010	010	101_2
010	010	101_2	2	2	5_8

The above relationship of octal to binary simplifies the programming of digital computers, since the octal system may be used in place of the more cumbersome binary system, which is the actual language of digital computers. The conversion from octal to binary and vice versa is, then, a simple process which may be accomplished at any point in the system as desired.

Hexadecimal Systems

The hexadecimal system has a radix of 16. The ten digits of the decimal system and the first six letters of the alphabet are the symbols most commonly used to represent the 16 digits of the hexadecimal system. (See TABLE 1-1.) The number 16, like 8, is an integral power of two (2). That is, $16_{10} = 2^4_{10}$. Thus, one hexadecimal digit has a value equivalent to that of a group of four binary digits and vice versa, as indicated below.

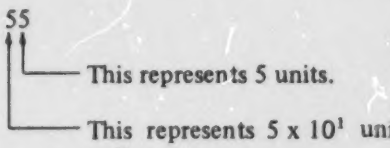
Hexadecimal to Binary			Binary to Hexadecimal		
E	2	5_{16}	1110	0010	0101_2
1110	0010	0101_2	E	2	5_{16}

POSITIONAL NOTATION

The radix or base of a number system was previously described as the total number of symbols or characters used to count from zero to the radix minus one or (R-1) of that number system. Therefore, once R-1 has been reached and you wish to exceed the radix, you must have a method or procedure for doing this. Without a procedure to follow, you will end up

with the same problem that the Hottentot had: "One, two, ... , plenty." One procedure for doing this involves using positional notation.

Positional notation is a system where the value or magnitude of a number is defined not only by its symbol, but also by its position. Examine the number 55 to see this illustrated.



Each position in the positional notation system represents a power of the radix or base, and is ranked in ascending or descending order. Magnitude can be extended by adding symbols to the 55. To see this demonstrated, examine the following line graph:

10^2	10^1	10^0	10^{-1}	10^{-2}	10^{-3}
5	5	5	5	5	5

This number in reality is

$5 \times 100 + 5 \times 10 + 5 \times 1 + 5 \times 0.1 + 5 \times 0.01 + 5 \times 0.001$

or 555.555

The above number is shown in base 10 (radix of 10), but it could just as easily have been in some other base. In a positional number system, the radix of the system is always written as a subscript to the number written. For instance, 55 in base 8 is 55_8; 55 in base 16 is 55_{16}; and 55 in base 10 is 55_{10}. However, base 10 numbers are written by convention without a subscript, except when they are used in conjunction with other number systems, i.e., $5_8 + 7_{10} + 101_2 + 9_{16}$ . . . etc.

The four number systems that you will study in this topic are positional notation systems. This does not mean that all number systems are positional notation systems. If you look on the back of a dollar bill, you will find the Roman number MDCCLXXVI. The equivalent number in the decimal number system is 1776. The Roman number system is not a positional number system. In the Roman system each number consists of a group of characters, each of which is distinct unto itself. The total value of the number is obtained by adding the sum of all the separate symbols. This is demonstrated by the following horizontal line graph:

M	D	C	C	L	X	X	V	I	
1000	+ 500	+ 100	+ 100	+ 50	+ 10	+ 10	+ 5	+ 1	= 1776

Notice that the position of each character does not affect its value, and that no value is less than one (1).

The decimal system, on the other hand, is a positional notation system. Therefore, the number 1776 is expressed this way:

10^3	10^2	10^1	10^0
1000	100	10	1
x	x	x	x
1	7	7	6

10^0 is the 1's column X6 = 6

10^1 is the 10's column X7 = 70

10^2 is the 100's column X7 = 700

10^3 is the 1000's column X1 = 1000

1776

You can write larger numbers simply by adding an additional power of 10 column. For example, if you want to increase the magnitude of 1776 by 90,000, simply write

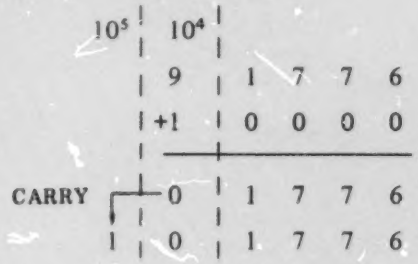
10^4	10^3	10^2	10^1	10^0
9	1	7	7	6

In this example you increased the value of 1776 by 90,000 by placing the symbol for nine in the 10^4 position, or

$9 \times 10{,}000 + 1776 = 91{,}776$

Look at the 9 in the 10^4 position. In this example, 9 is the largest symbol that can be

written in the 10^4 column. Now, if you add 10,000 to 91,776 the number in the 10^4 position will read the value of the radix (10×10^4). At this point a carry is generated from the 10^4 position to the 10^5 position.

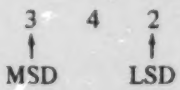


The number is now 101,776 or

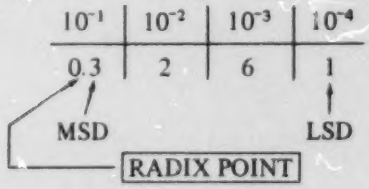
$1 \times 10^5 + 0 \times 10^4 + 1 \times 10^3 + 7 \times 10^2 + 7 \times 10^1 + 6 \times 10^0$

The zero in the above example indicates a place holder. While zero has no value in itself ($0 \times 10 = 0$), it does raise the magnitude of the number by one power of ten ($10^4 \rightarrow 10^5$).

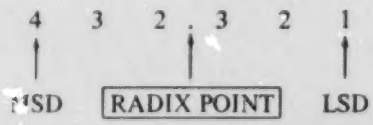
Now look at decimal number 342. The digit in this number (342) that would change the value of the number least, if changed, is 2. The term which describes the digit located in this position is called the Least Significant Digit or LSD; the digit that would change the value of the number 342 most, if changed, is called the Most Significant Digit or MSD. The MSD is always the leftmost nonzero digit.



This also applies to fractional numbers less than one (1), e.g.,



and to mixed numbers, e.g.,



While decimal numbers are used in the preceding examples to explain positional notation, the principles examined apply to all positional notation number systems. Take, for example, the octal system (base 8). As in the decimal system the largest symbol that can be written in one position is R-1, or

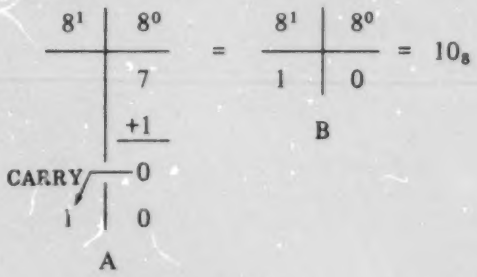
Radix = 8

R - 1 = 7

8 - 1 = 7

Seven, therefore, is the largest symbol in the base 8 system.

If one unit is added to an R-1 symbol, a carry is generated into the next highest position. The following illustrates the CARRY PRINCIPLE.



In part A, 7 is equal to R-1. When one is added to 7, a carry is generated into the 8^1 position, as shown in parts A and B. As you can see, octal 7 has been changed to octal 10 (See note below.). In other words, 10 in the octal system is equivalent to 8 in the decimal system.

NOTE: The symbol "10" (pronounced one zero) always represents the radix in its own system. This is true because the radix is one unit larger than the largest character, and by the rules of counting, this value is written as "10."

For example:

Binary "10" = two (the radix of the binary system)

Octal "10" = eight (the radix of the octal system)

Decimal "10" = ten (the radix of the decimal system)

Hexadecimal "10" = sixteen (the radix of the hexadecimal system)

The binary system is operated in the same manner, as shown below.

Radix = 2

R - 1 = 1

2 - 1 = 1

R - 1 + 1 = 2^1 | 2^0 = 2^1 | 2^0 = 10_2

	2^1	2^0
	1	
+1		
carry	0	
	1	0

CONVERSION TO DECIMAL

The similarities between the binary, octal, decimal, and hexadecimal number systems should be apparent. You will follow five steps in order to convert a number in any positional notation system to a number in the decimal system:

Step 1—Count the number of symbols. This number will be equal to the number of positions needed; i.e., 101101_2 = 6 positions.

Step 2—Write out the powers of the radix starting at zero and continuing until the number of positions obtained in step 1 is reached.

2^5	2^4	2^3	2^2	2^1	2^0

Step 3—Write the numbers under the columns from step 2, placing the LSD under the 2^0 position.

2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	1	0	1

Step 4—Multiply the symbol in each column by the power of the radix for each column in which the symbol appears.

	2^5	2^4	2^3	2^2	2^1	2^0
Step 2–						
	x	x	x	x	x	x
Step 3–	1	0	1	1	0	1
Step 4–	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
	1 x 32	0 x 16	1 x 8	1 x 4	0 x 2	1 x 1
	32	0	8	4	0	1

Step 5—Add up all of the products.

$$32 + 0 + 8 + 4 + 0 + 1 = 45_{10}$$

This same method can be applied to octal numbers. Take, for example, the number 346_8. To find its decimal value, follow the same procedure:

Step 1. 346_8 = 3 positions.

Step 2. Write out the positions.

8^2	8^1	8^0

Step 3. Fill in the numbers.

8^2	8^1	8^0
3	4	6

Step 4. Multiply each number by the power of the radix in its column.

$8^2 = 64$	$8^1 = 8$	$8^0 = 1$
3 x 64	4 x 8	6 x 1
192	32	6

Step 5. Add all the products from step 4.

$$192 + 32 + 6 = 230$$

Therefore, $346_8 = 230_{10}$

Q1. What is the decimal value of each of the following numbers?

1. 100101_2
2. 5761_8
3. 6710_8
4. 10010010_2

The procedure you followed to convert the numbers in the above problems was a bit cumbersome. A quicker and easier procedure is shown below. To convert octal number 3481 to a decimal number, follow these six steps:

Step 1–Write out the number.

$3\ 4\ 8\ 1_8$

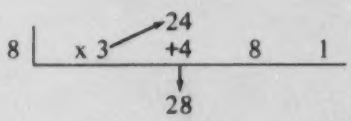
Step 2–Place the number inside a half box.

$\lfloor 3\ 4\ 8\ 1$

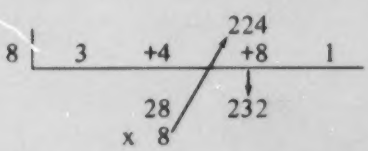
Step 3–Place the radix outside the box.

$8\lfloor 3\ 4\ 8\ 1$

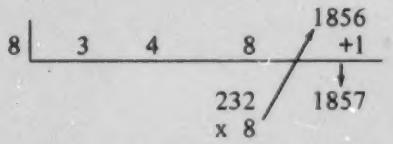
Step 4–Starting at the MSD, multiply the first number by the radix and add the product to the next number.



Step 5–Multiply the sum obtained in step 4 by the radix and add the product to the next number.



Step 6–Multiply the sum obtained in step 5 by the radix and add the product to the next number, which is the LSD.



Thus, $3481_8 = 1857_{10}$

Once you have reached the LSD, simply add the product from the previous number to the LSD. Do not multiply further, because this is your answer.

The same steps can be used to convert a binary number to a decimal number. To convert 10010_2 to base 10 (decimal), for example, proceed as follows:

Step 1–Write out the number.

10010_2

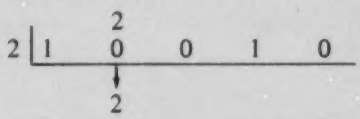
Step 2–Draw the half box.

$\lfloor 10010$

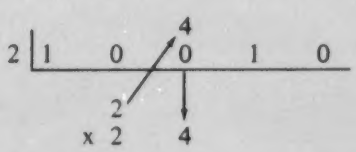
Step 3–Place the radix outside the box.

$2\lfloor 10010$

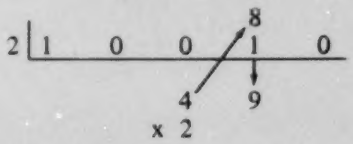
Step 4–Multiply the MSD by the radix and add the product to the next digit.



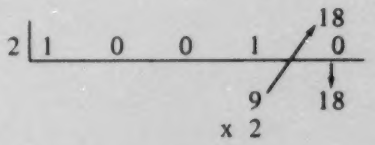
Step 5–Multiply the sum obtained in step 4 by the radix and add to the next digit.



Step 6–Multiply the sum obtained in step 5 by the radix and add to the next digit.



Step 7–Multiply the sum obtained in step 6 by the radix and add to the LSD.



Step 8–STOP! This is your answer.

$10010_2 = 18_{10}$

Q2. **Using the above method, convert the following numbers to base 10 numbers.**

1. 73_8
2. 10110_2
3. 512_8
4. 36_8
5. 111011101_2

CONVERSION TO OCTAL FROM BINARY

Converting the number in question 2, problem 5, above requires considerable time to do correctly. If you compare the time you spent converting binary number 111011101 to decimal number 477 with the time spent by a binary computer to process a binary number that has a decimal value in the billions, you can see that a straight binary-to-decimal conversion system for large numbers is not very practical. For this reason, the octal number system is used in computer technology. The octal system is used as an intermediate system between the decimal and binary number systems. Using the intermediate system, a computer can carry out computations in binary, the answers can be easily converted to octal, and then to decimal. This is possible because both the binary and octal systems generate a carry when going from seven to eight. This will be demonstrated in the next paragraph for numbers 1_8 through 7_8 and 1_2 through 111_2. The two systems are equivalent, and are directly convertible.

To see this, first count in binary from one to eight. Then advance the count by starting with binary one and adding one to it until binary eight is reached. Remember that a carry is generated each time that the radix (2) of the binary number system is reached.

BINARY		OCTAL	
Binary 0 = 000		Octal 0 = 0	
Add one +1		Add one +1	
Binary 1 = 001		Octal 1 = 1	
Add one +1	generate a carry	Add one +1	
Binary 2 = 010		Octal 2 = 2	
Add one +1		Add one +1	
Binary 3 = 011		Octal 3 = 3	
Add one +1	generate a carry	Add one +1	
Binary 4 = 100		Octal 4 = 4	
Add one +1		Add one +1	
Binary 5 = 101		Octal 5 = 5	
Add one +1	generate a carry	Add one +1	
Binary 6 = 110		Octal 6 = 6	
Add one +1		Add one +1	
Binary 7 = 111		Octal 7 = 7	
Add one +1	generate three carries	Add one +1	generate a carry
Binary 8 = 1000_2		Octal 8 = 10_8	

At 8, in both systems, a carry is generated. In octal the radix is reached at 8; this same point in binary also generates a carry. If three binary

A1. 1. 37

2. 3057

3. 3528

4. 146

A2. 1. 59

2. 22

3. 330

4. 30

5. 477

---

digits are used to express each octal digit, the two systems are equal and directly convertible.

To convert from binary to octal, you will use the same procedure. First, write 8 in binary as 1000 and place the binary digits in groups of three starting from the LSD. Next, use table 1-2 with each group of three binary digits in the example, and select the octal equivalent. Write the octal equivalent below each group of binary digits as shown below.

MSD	LSD	
1	000	binary 8
↓	↓↓↓	
1	0	octal 10

In the example below, this procedure is used to convert binary 10110011101101 to its equivalent octal number.

10	110	011	101	101_2
2	6	3	5	5_8

Note that when the MSD position is reached, if there is not a group of three binary digits to express the octal equivalent, you simply add zeros ahead of the MSD, as shown in the following example.

010	110	011	101	101_2
2	6	3	5	5_8

This will make the number of binary digits evenly divisible by three without affecting the value of the number.

Table 1-2.—Binary to Octal Conversion

BINARY		OCTAL
000	=	0
001	=	1
010	=	2
011	=	3
100	=	4
101	=	5
110	=	6
111	=	7

CONVERSION TO BINARY FROM OCTAL

It is just as easy to go from octal to binary. Simply write the three binary equivalent numbers under each octal digit. This is shown below using the octal number 43210_8.

43210_8 =	4	3	2	1	0
100011010001000_2 =	100	011	010	001	000

It will be to your advantage to memorize the digits comprising binary numbers 1 through 7, especially if you plan to continue in computer technology, since you will use the octal system

extensively. However, unless you have memorized these binary numbers, you should refer to table 1-1 in answering the following comprehension questions:

Q3. What is the octal value of each of the following binary numbers?

1. 1011101110111101
2. 101111111101101
3. 1011000011101011
4. 1111110111011

Q4. Convert the following binary numbers to octal, and then to decimal.

1. 101101101
2. 10110011000111
3. 0111011000001
4. 1111101
5. 111001101010111

CONVERSION FROM DECIMAL TO OCTAL AND BINARY

All the conversions from base 2 and base 8 to base 10 were made by multiplying and adding. The procedure for converting base 10 numbers to base 8 and base 2 numbers is just the opposite. It is done using division and subtraction. This conversion procedure is illustrated below.

Problem: Convert 105_{10} to its binary and then to its octal equivalent.

Step 1–Set up the problem for division.

$$\overline{)105}$$

Step 2–Divide the base 10 number by the radix of the system to which you wish to convert.

$$2\overline{)105}$$

Step 3–Divide the base 10 number by the radix, and extract the remainder from each step. The remainder that comes off first will be the LSD; the remainder that comes off last will be the MSD. Continue the division by dividing the quotient obtained by the radix until the dividend becomes smaller than the divisor. At this point, the dividend is the remainder and is the MSD, as illustrated below.

$$\begin{array}{r} 52 \\ 2\overline{)105} \\ 10 \\ \hline 5 \\ 4 \\ \hline 1 \end{array} \longrightarrow 1 \quad \text{LSD}$$

$$\begin{array}{r} 26 \\ 2\overline{)52} \\ 4 \\ \hline 12 \\ 12 \\ \hline 0 \end{array} \longrightarrow 0$$

$$\begin{array}{r} 13 \\ 2\overline{)26} \\ 2 \\ \hline 6 \\ 6 \\ \hline 0 \end{array} \longrightarrow 0$$

$$\begin{array}{r} 6 \\ 2\overline{)13} \\ 12 \\ \hline 1 \end{array} \longrightarrow 1$$

$$\begin{array}{r} 3 \\ 2\overline{)6} \\ 6 \\ \hline 0 \end{array} \longrightarrow 0$$

$$\begin{array}{r} 1 \\ 2\overline{)3} \\ 2 \\ \hline 1 \end{array} \longrightarrow 1$$

$$\begin{array}{r} 0 \\ 2\overline{)1} \\ 0 \\ \hline 1 \end{array} \longrightarrow 1 \quad \text{MSD}$$

A3. 1. 135675_8

2. 57755_8

3. 130353_8

4. 17673_8

A4.

1.	555_8	365_{10}
2.	26307_8	11463_{10}
3.	7301_8	3777_{10}
4.	175_8	125_{10}
5.	71527_8	29527_{10}

Step 4–Write out the number, from the MSD to the LSD.

$$1101001_2$$

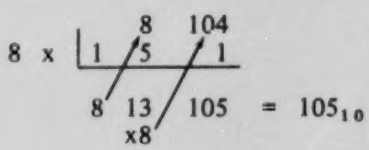
Now, to convert binary number 1101001 to its octal equivalent, you will use the conversion procedure demonstrated earlier. That is,

1	101	001
1	5	1

$= 151_8$

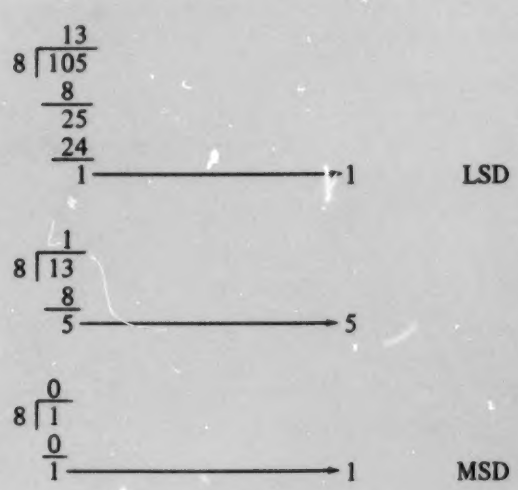
Thus, $105_{10} = 151_8 = 1101001_2$

To prove your answer, simply convert the octal number back to decimal.

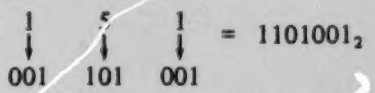


$= 105_{10}$

Going from decimal to binary is simple since it involves division by two. It does, however, have its drawbacks. Division by two is a time consuming process. A quicker method is to convert to octal, then to binary. This means you start with the decimal number 105, convert it to octal, and then to binary, as shown below.



Therefore, $105_{10} = 151_8$



$= 1101001_2$

Now that you have seen the conversion procedures, a little practice will let you master them. Here are some practice questions.

Q5. Convert the following numbers to decimal.

1. 1011101_2
2. 1567_8
3. 11010_8
4. 101156_8
5. 11101101101_2

Q6. Convert the following decimal numbers to octal and then to binary.

1. 101
2. 12
3. 5672
4. 11010
5. 328

CONVERSION OF FRACTIONAL NUMBERS

Fractional numbers are converted in much the same manner as whole numbers. Fractional numbers are arranged in order of the descending negative power of the radix. For example,

	10^{-1}	10^{-2}	10^{-3}	10^{-4}
	1/10	1/100	1/1000	1/10,000
	0.1	0.01	0.001	0.0001
MSD LSD 0.2173 =	2 x 0.1	+ 1 x 0.01	+ 7 x 0.001	+ 3 x 0.0001
RADIX POINT	+ .2	+ 0.01	+ 0.007	+ 0.0003
	0.2	1	7	3

CONVERTING BINARY FRACTIONS TO OCTAL FRACTIONS

When a binary number is converted to an octal number, the negative power of the radix has no effect because of the straight conversion. The conversion procedure you use is the same, with the following exception: group the binary digits in groups of three starting from the MSD, not from the LSD. The reason for this is: if the number of binary digits is not evenly divisible by three, the addition of zeros after the LSD will not affect the value of the number. For example,

	MSD ↓			
0.100101101100_2 =	0.100	101	101	100
	0.4	5	5	4_8

or

MSD ↓

$0.100101101100_2 = 0.4554_8$

CONVERTING OCTAL FRACTIONS TO DECIMAL FRACTIONS

When converting from octal to base 10, you must compensate for the expression of the negative powers. This is easy to do. Simply ignore the negative power and treat the number as a whole number. Convert the octal fraction to a decimal fraction using the same procedure as before. Once the number is converted, divide it by the conversion factor. The conversion factor is equal to the positive power of the radix at whose position the fractional LSD appears. The conversion factor is obtained as follows:

Step 1 – Write the octal fraction.

0.4554_8

Step 2 – Determine the negative power of the radix (8, in this case), at whose position the LSD appears.

8^{-1}	8^{-2}	8^{-3}	8^{-4}
0 .4	5	5	4
			↑ LSD

Now that you know the LSD is at the 8^{-4} position, disregard the negative sign and raise the base (8) to the fourth power.

$8^4 = 4096_{10}$

You now have the conversion factor, which is used to convert octal 0.4554 to its decimal equivalent.

Step 1 – Disregard the decimal point in 0.4554 and write the octal number as if it were a whole number.

4554_8

Step 2 – Set up the octal number for conversion to base 10, as previously explained.

4554_8 = 4 positions

Step 3 – Write out the positions.

8^3 | 8^2 | 8^1 | 8^0

A5. 1. 93

2. 887

3. 4616

4. 33390

5. 1901

A6. 1. 145_8 1100101_2

2. 14_8 1100_2

3. 13050_8 1011000101000_2

4. 25402_8 10101100000010_2

5. 510_8 101001000_2

Step 4—Fill in the numbers.

8^3	8^2	8^1	8^0
4	5	5	4

Step 5—Multiply each number by the power of the radix in its column.

$$4 \times 8^3 + 5 \times 8^2 + 5 \times 8^1 + 4 \times 8^0$$

$$4 \times 512 + 5 \times 64 + 5 \times 8 + 4 \times 1$$

$$2048 + 320 + 40 + 4$$

Step 6—Add all the products from step 5.

$$2048 + 320 + 40 + 4 = 2412_{10}$$

Step 7—Divide the product obtained in step 6 by the conversion factor you obtained earlier.

$$\frac{2412}{4096} = 0.58888$$

Thus, the decimal equivalent of 0.4554_8 is 0.58888.

Q7. What is the octal to decimal conversion factor for each of the following numbers?

1. 0.01101101101111_2
2. 0.713712_8
3. 0.12_8
4. 0.111011101_2

CONVERTING DECIMAL FRACTIONS TO OCTAL AND BINARY FRACTIONS

It is easier to convert a decimal fraction to an octal fraction and to a binary fraction than it is to convert an octal fraction to a decimal fraction. Instead of using a conversion factor, reverse the procedure you learned for converting decimal to octal. To convert a decimal fraction to an octal fraction and to a binary fraction, simply multiply and subtract in that order. That is first multiply the decimal fraction by the radix of the number system to which you wish to convert it. Next, subtract everything that appears to the left of the radix (decimal) point. When a fraction is converted, the MSD will come off first. Thus, to convert decimal number 0.589 to an octal number, proceed as follows:

```
          0.589
        x     8
MSD .4 <- 4.712
          0.712
        x     8
   5 <-   5.696
          0.696
        x     8
   5 <-   5.568
          0.568
        x     8
LSD 4 <-  4.544
```

Thus, $0.589_{10} = 0.4554_8$

This conversion may be carried out to as many places as needed, but generally four places are

enough. Octal fraction 0.4454_8 can now be broken down into a binary number:

0.4	5	5	4_8
↓	↓	↓	↓
0.100	101	101	100_2

Q8. Convert each of the following decimal fractions to a four-place octal fraction; then convert each to a twelve-place binary fraction.

1. 0.587
2. 0.987
3. 0.642
4. 0.2017
5. 0.9716

CONVERSION OF MIXED NUMBERS

Mixed numbers consisting of whole numbers and fractions are easy to convert from one number system to another. Just split the mixed number in half at the radix point, then convert both halves in the manner you have previously been shown. Take, for example, the problem below:

Convert 1101111.1101_2 to octal and decimal.

Step 1—Split the number at the radix point.

$$1101111 \quad . \quad 1101_2$$

Step 2—Decide on which conversion procedure to use, that is, either to convert from binary to decimal to octal or from binary to octal to decimal. Both procedures contain the same steps. In this example, the binary to octal to decimal procedure is used.

Step 3—Convert the binary number to octal (see table 1-1).

1	101	111	.	110	100	← Two zeros added to the LSD of the fraction to make it evenly divisible by three.
1	5	7	.	6	4	

Thus, $1101111.1101_2 = 157.64_8$

Step 4—Split the octal number into two parts at the radix point.

$$157 \quad . \quad 64_8$$

Step 5—Convert the left half to decimal.

$$8 \overline{)\ 157} = 111_{10}$$

Step 6—Convert the right half to decimal.

$$8 \overline{)\ 64} = 52_{10}$$

Step 7—Apply the conversion factor 64.

$$\frac{52}{64} = 0.8125_{10}$$

Step 8—Add the two halves together.

$$\begin{array}{r} 111 \\ +\ 0.8125 \\ \hline 111.8125 \end{array}$$

Thus, $1101111.1101_2 = 157.64_8$ or 111.8125_{10}

Now, to convert from a decimal mixed number to binary or octal, use the same procedure. Convert, for example, decimal number 111.8125 to octal to binary.

Step 1—Split the mixed number into two parts.

$$111 \quad . \quad 8125_{10}$$

A7. 1. 32768

2. 262144

3. 64

4. 512

A8. 1.	0.4544_8	0.100101100100
2.	0.7713_8	0.111111001011
3.	0.5106_8	0.101001000110
4.	0.1472_8	0.001100111010
5.	0.7614_8	0.111110001100

Step 2–Convert each half to octal.

```
      13
8 ) 111
     8
    31
    24
     7   ← remainder → 7 LSD

       1
8 ) 13
      8
      5   ← remainder → 5

       0
8 ) 1
      0
      1   ← remainder → 1 MSD

111 = 157₈
```

```
0.8125
x     8
6.5000                    6 MSD
  └──────────────────────┘

0.5000
x     8
4.0000                    4 LSD
  └──────────────────────┘

0.8125 = 0.64₈
```

Step 3–Add the two halves together.

$$\begin{array}{r} 157 \\ +\ 0.64_8 \\ \hline 157.64_8 \end{array} = 1\ 5\ 7\ .\ 6\ 4_8$$

Step 4–Convert to binary.

1 101 111 . 110 100

Thus, $111.8125_{10} = 157.64_8 = 1101111.1101_2$

Q9. **Convert the following numbers to the indicated numbering system.**

1. 1507.06_8 to binary
2. 1101000111.000110_2 to decimal
3. 777.77_8 to binary to decimal
4. 10111.101101_2 to decimal
5. 6366.36_8 to decimal
6. 983.983 to octal
7. 7106.532 to binary

CONVERSION BETWEEN HEXADECIMAL AND BINARY

The number 16, like 8, is also an integral power of 2. As previously indicated, grouping may be used to convert from binary to hexadecimal and vice versa. With one exception, the procedure you use is the same as that used for binary to octal conversion. The exception is that when converting from binary to hexadecimal, you divide the binary number into groups of four digits instead of three, and assign the hexadecimal equivalent to each group.

Problem: Convert 11011.01101_2 to its hexadecimal equivalent.

Step 1–Group the digits.

11 1011.0110 1

Step 2–Add extra zeros.

0011 1011.0110 1000

Step 3—Assign the hexadecimal equivalent from table 1-1

0011 1011.0110 1000

3 B . 6 8

Thus, $111011.01101_2 = 3B.68_{16}$

When converting from hexadecimal to binary, assign a binary equivalent to each hexadecimal digit.

Problem: Convert $3C8.96_{16}$ to its binary equivalent.

Assign binary equivalents from table 1-1.

3 C 8 . 9 6

0011 1100 1000 . 1001 0010

Thus, $3C8.96_{16} = 1111001000.1001011_2$

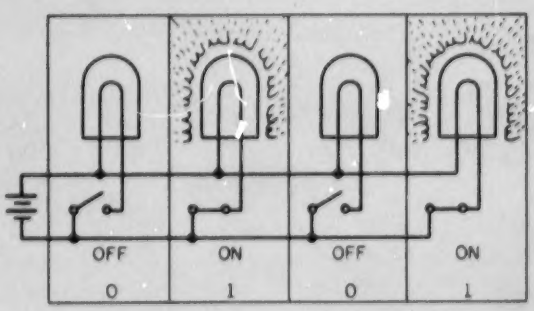
SUMMARY OF NUMBER SYSTEMS

With the completion of this topic, you should have gained a basic understanding of Number System Concepts. The number systems which were dealt with are used quite extensively in the digital computer field. The following is a summary of the important points in the topic, Number Systems.

BINARY	OCTAL	DECIMAL
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	10	8
1001	11	9
1010	12	10
1011	13	11
1100	14	12
1101	15	13
1110	16	14
1111	17	15
10000	20	16

RADIX—The radix is the base of a positional number system. It is equal to the number of digits or symbols needed to count from zero to the radix, or base minus one, of that number system.

BINARY NUMBER SYSTEM—The components used in a computer can assume only one of two possible conditions. The two conditions are conducting or not conducting. Binary 1, or TRUE, and 0, or FALSE, and vice versa, are used to represent these two states. The binary system, having a base 2, is therefore a natural choice.



A9. 1. 1101000111.000110_2

2. 839.09375_{10}

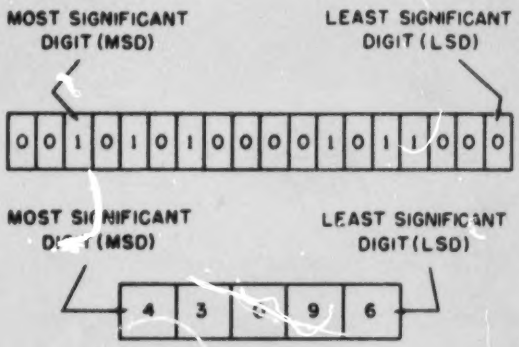
3. 11111111.111111_2 and 511.9844_{10} (rounded off)

4. 23.703125_{10}

5. 3318.46875_{10}

6. 1727.7672_8

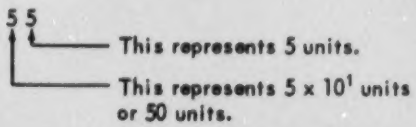
7. $1101111000010.100010000011_2$


MOST SIGNIFICANT DIGIT (MSD)
LEAST SIGNIFICANT DIGIT (LSD)
0 0 1 0 1 0 1 0 0 0 0 1 0 1 1 0 0 0
MOST SIGNIFICANT DIGIT (MSD)
LEAST SIGNIFICANT DIGIT (LSD)
4 3 9 6

MOST SIGNIFICANT DIGIT (MSD)–The MSD is the digit whose position within a given number expression has the greatest weighting power.

LEAST SIGNIFICANT DIGIT (LSD)–The LSD is the digit whose position within a given number expression has the least weighting power.

POSITIONAL NOTATION–It is a system in which the value or magnitude of a number is defined not only by its digits or symbol value, but also by its position. Each position represents a power of the radix or base, and is ranked in ascending or descending order.



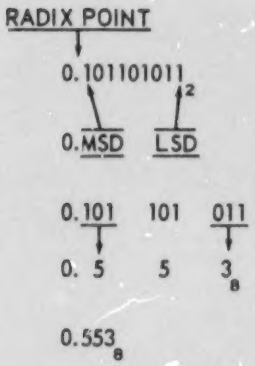
OCTAL NUMBER SYSTEM–The Octal system (base 8) is quite useful as a tool in the conversion of binary numbers. This system works because 8 is an integral power of 2, that is, $2^3 = 8$. It is an easy matter to convert from base 2 to base 8, and then to base 10. The use of octal numbers reduces the number of digits required to represent the binary equivalent of a decimal number.

$$100011010001000_2 = \begin{array}{c|c|c|c|c} 100 & 011 & 010 & 001 & 000 \\ 4 & 3 & 2 & 1 & 0 \end{array} = 43210_8$$

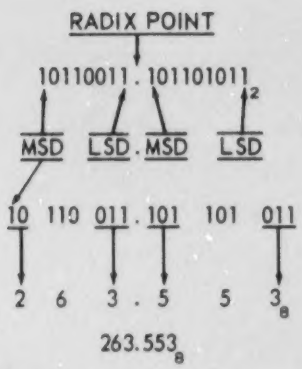
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Hexadecimal to Binary			Binary to Hexadecimal		
E	2	5	1110	0010	0101
1110	0010	0101	E	2	5

HEXADECIMAL NUMBER SYSTEM–The hexadecimal number system (base 16) is sometimes used in computer systems. A binary number can be converted directly to a base 16 number if the binary number is first broken into groups of four digits.



CONVERSION OF FRACTIONAL NUMBERS–Fractional numbers are converted from one number system to another in much the same manner as are whole numbers. Fractional numbers are arranged in order of the descending negative powers of the radix. Remember that the MSD is the first digit to the right of the radix point.



CONVERSION OF MIXED NUMBERS–Mixed numbers consisting of whole numbers and fractions can be converted from one number system to another. Split the mixed number at the radix point, then convert each half.

TOPIC 2

BOOLEAN ALGEBRA

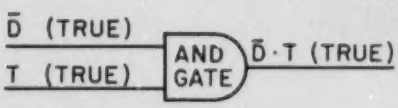
Boolean logic was devised by George Boole in the nineteenth century, before the development of electronics. Boolean logic, or algebra as it is now called, is used in modern computer technology because it is based on the logical assumption that most quantities have two possible states—"TRUE" or "FALSE." This can be applied in the old adage "she loves me, she loves me not." If "she loves me" is TRUE, then "she loves me not" must be FALSE. The opposite state in Boolean logic is equally possible, where "she loves me not" is TRUE and "she loves me" is FALSE. This illustrates the point that, depending upon the condition that is set, every logic term has two states, one TRUE and one FALSE. These opposite states are called COMPLEMENTS; thus, "she loves me not" is the complement or opposite of "she loves me." When either state is TRUE, its complement is FALSE.

THE AND GATE

Boolean is nothing more than a description of the input conditions necessary to get a desired output from a logic circuit. To further illustrate this, let's use an example very familiar to you, "Going on Liberty." Some of the conditions that you will have to meet before you are actually on your way are described in the following paragraphs.

Being "ON LIBERTY" will be considered a TRUE statement, while its complement "LIBERTY NOT" will be considered a FALSE statement. These conditions can also be stated with LIBERTY equaling L, and its complement, LIBERTY NOT, equaling $\overline{L}$. The bar or vinculum above the letter L signifies that it is the complement of L. With these conditions, we have either LIBERTY = L or LIBERTY NOT = $\overline{L}$. If L (liberty) is TRUE and is the output of the logic circuit, what inputs are necessary to get you on liberty?

First, you must NOT have the duty. If duty is represented by D, then duty NOT is $\overline{D}$. When time is considered, you know that it must be after liberty call. This time factor can be represented by T, and time before liberty call-its complement-by $\overline{T}$. Liberty, then, depends on two conditions being true: no duty, or $\overline{D}$, and liberty call, or T. Instead of using many words to describe the condition of liberty, it can be put into Boolean shorthand as $\overline{D} \cdot T$ = Liberty and logically diagramed as



$\overline{D}$	T	$\overline{D} \cdot T$
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

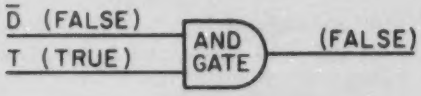
TRUTH TABLE

NOTE: The use of D, T, and L as variables is not to be interpreted as a constant. Any letter, or letters, may be substituted in variable representation.

The preceding Boolean expression, TRUTH TABLE and related logic diagram are for the AND gate shown above. In an AND gate ALL inputs must be TRUE to get a TRUE output (refer to the truth table). The AND function is

indicated by a dot (·) between terms, or simply by grouping the terms together. Thus $\bar{D}T$ and $\bar{D} \cdot T$ are both AND function expressions and are read as "Dee not and Tee." If $\bar{D}T$ (duty NOT and time) is TRUE, you go on liberty, as shown by the truth table.

The other condition is that you can not go on liberty. This may be because you have the duty or it is not time for liberty call or both. These three conditions can be logically diagramed as

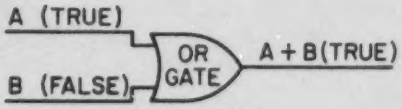


It is apparent from the truth table that if either input condition is FALSE, the output condition will be FALSE. Therefore, assuming that $\bar{D}$ is FALSE (you have the duty) and T is TRUE (liberty call), then the output is FALSE and you can not "go on liberty" even though liberty call has gone because you have the duty.

THE OR GATE

There is still a way that you can "go on liberty." However, you must first learn another logic function–the OR gate. The OR gate is indicated by the + sign between terms and logically diagramed as

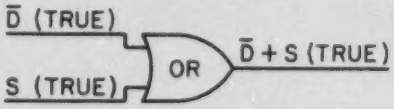
The OR function differs from the AND function in that only ONE input need be TRUE to get a TRUE output, as in the following logic diagram.



A	B	A + B
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

Now, go back to the immediate problem, which is that $\bar{D}$ is FALSE and that you have the duty. To meet the conditions necessary for liberty, the $\bar{D}$ input to the AND gate must be changed to a TRUE state. To change your status from one of duty to one of no duty, you need a standby (S). This can be expressed and diagramed using an OR gate as shown in the following example.

$\bar{D} + S$ = NO DUTY



$\bar{D}$	S	NO DUTY
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

This OR function takes care of the "no duty" requirement by giving you a no duty status any time $\overline{D}$ or S is TRUE. This OR function can be combined with the AND function (in which T was TRUE); this will then make liberty TRUE. One way to do this is to draw the logic diagram for both expressions ($\overline{D}T$ and $\overline{D} + S$), combining them as shown in figure 2-1. From the combined gate we can now write the total Boolean expression.

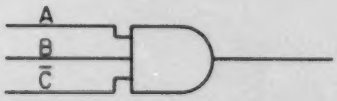
$$T(\overline{D} + S) = \text{Liberty}$$

$\overline{D} + S$ is enclosed in parentheses to show that the $\overline{D}$ and S variables were processed through a common gate, and T is written next to the parentheses to indicate that it is ANDed with the output of the common gate.

If additional grouping signs are necessary for an expression that already contains parentheses, use brackets.

Q1. When $\overline{X}$ is TRUE what is the state of X?

Q2. What is the Boolean output expression for the following logic diagram?



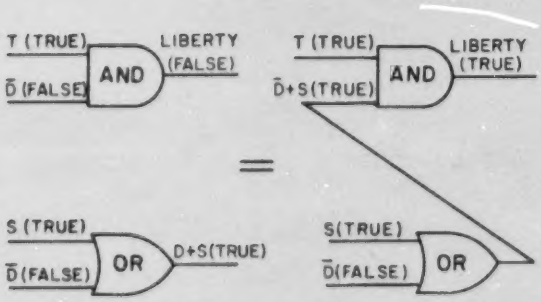
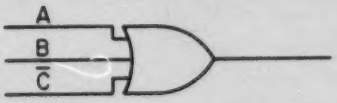


Figure 2-1.—The Combined AND and OR function.

Q3. If $\overline{C}$ is FALSE, what is the output of the gate in Q2?

Q4. Write the Boolean output expression for the following logic diagram.

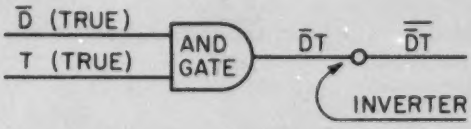


Q5. In question 4, if A and B are each FALSE and $\overline{C}$ is TRUE, what is the output?

Q6. If the conditions for A and B as stated in question 5 are the same, but $\overline{C}$ changes to a FALSE state, what is the output?

There are only two additional gates, and their Boolean expressions, which you need to learn before you are ready to tackle Boolean algebra. These are the NAND and the NOR gates. Actually, these gates are nothing more than AND and OR gates with an inverter on their OUTPUTS. The inverter is logically diagramed as a circle. Any Boolean expression inputted to an inverter is outputted in the opposite state. For example, if you have liberty, $\overline{D}T$ = TRUE, applied to an inverter, the output will be inverted and will become $\overline{D}T$ = FALSE, or $\overline{\overline{D}T}$

This is logically diagrammed as

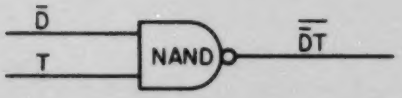


As you can see, you went from the liberty status to the opposite or liberty not status

(maybe you "stood by" for our earlier example).

THE NAND GATE

The NAND function can be used to express this inverted condition and is logically diagramed as



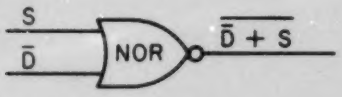
$\bar{D}$	T	$\overline{\bar{D}T}$
FALSE	FALSE	TRUE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

TRUTH TABLE

The output $\overline{\bar{D}T}$ is read as "Bar D NOT, T." The vinculum (BAR) is used as a grouping symbol to indicate that $\bar{D}T$ was ANDed and inverted through a common NAND gate. The output of this NAND gate is $\bar{D}T$.

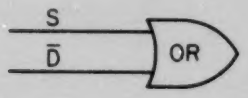
THE NOR GATE

The NOR gate functions in the same manner as a NAND gate, except that its parent gate is a basic OR gate with an inverter on its output. The logic diagram for the Boolean output expression $\overline{\bar{D} + S}$ is illustrated below.

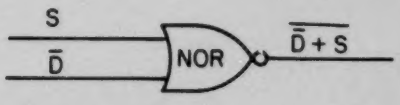


In drawing the logic diagram, the following steps are used:

1. The OR symbol (which is called the parent gate) is drawn and its inputs are labeled.



2. The inverter (indicated by the solid vinculum above both terms) is then drawn as a small circle on the output of the OR gate, and labeled as shown below. You now have the logic diagram and expression for a NOR gate.



$\bar{D}$	S	$\overline{\bar{D} + S}$
FALSE	FALSE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	FALSE

TRUE TABLE

For the sake of simplicity, the INVERTER for the NAND gate and the INVERTER for the NOR gate are each shown as part of a parent gate, rather than separate from it.

THE NOT FUNCTION

You can now see how the inverter is used to change the logic state of a given Boolean term.

A1. FALSE

A2. $AB\overline{C}$

A3. FALSE

A4. $A + B + \overline{C}$

A5. TRUE

A6. FALSE

When the inverter is coupled to either the AND gate output or the OR gate output, the NAND and NOR functions are created.

The most important thing to remember is that the NAND gate and the NOR gate will each provide a TRUE output only if the output of the parent gate of each is FALSE.

Figure 2-2 illustrates the NAND function. The parent gate is an AND gate with inputs A and B being TRUE. The NOR gate is illustrated in figure 2-3. Its parent gate is an OR gate with inputs of A = TRUE and B = FALSE.

In summary, the NAND gate will output a TRUE if any input is FALSE. The NOR gate will output a TRUE only if all inputs are FALSE.

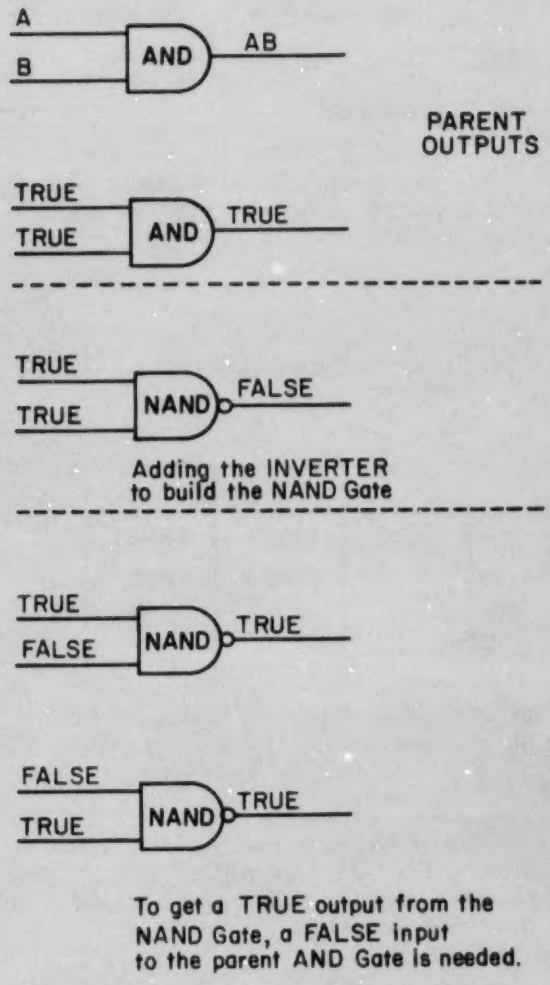


Figure 2-2.—The NAND Gate.

THE INPUT INVERTER

Sometimes it may become necessary to invert the INPUT logic to a gate. This is done by placing an INVERTER on the input side of the gate. The gate will still be an AND or an OR gate, but will have output expressions which are different from those that are normally obtained. Figure 2-4 illustrates logic gates with inverted inputs and their respective outputs.

In figure 2-4C, the NAND gate, the solid BAR (vinculum) above the output terms indicates the NAND function, while the split bars above the individual terms indicate an inverter on the input side of the gate. The same is true for figure 2-4D, the NOR function. It may be useful to point out at this time that, given a Boolean output expression containing an inverted function, or split vinculum (e.g., $A\overline{BD}$), it is impossible to tell, without the aid of a logic diagram, if the inverted function is inputted to

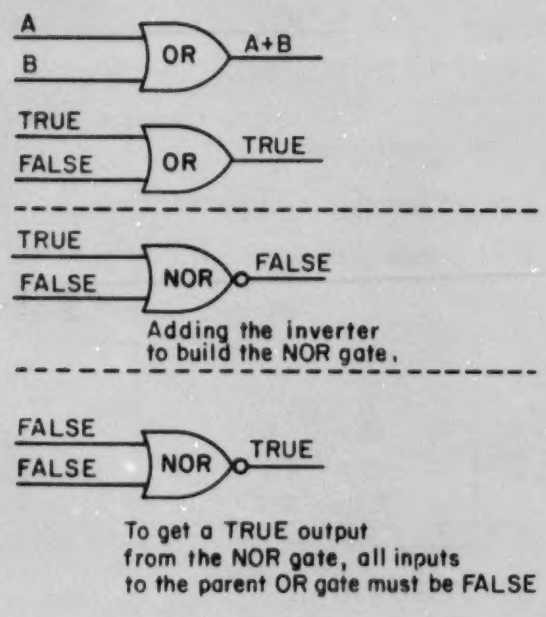


Figure 2-3.—The NOR Gate.

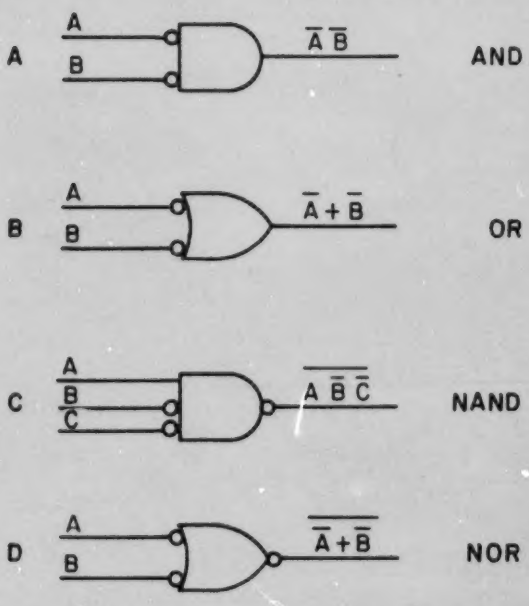
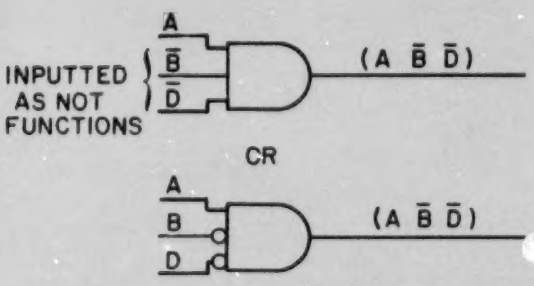


Figure 2-4.—The Input Inverter.

the logic gate as a NOT function or is the result of an inverter on the input.

Thus $A\bar{B}\bar{D}$ could be either



However, given a logic diagram, the correct Boolean expression can always be written. For example, the preceding logic diagrams will always be written as $A\bar{B}\bar{D}$.

As stated at the beginning of this topic, Boolean logic consists of only two possible states. Up to this point, we have designated these states as TRUE and FALSE.

It should be apparent that these two possible states, or conditions, could be represented by a counting system having only two numerals, for example, the binary system. If you let the TRUE state equal a one and the FALSE state equal a zero, you have such a system.

This binary system is used quite extensively in computer electronics.

Knowing this, there is no longer any need for us to deal in TRUE and FALSE conditions. You can now use a 1 to represent a TRUE, and a 0 to represent a FALSE.

REMEMBER! TRUE = 1 and FALSE = 0

The four basic logic circuits, each with its Boolean expression, may be summarized by the truth tables shown in figure 2-5. In the truth tables, we will consider a 1 as a TRUE state and a 0 as a FALSE state.

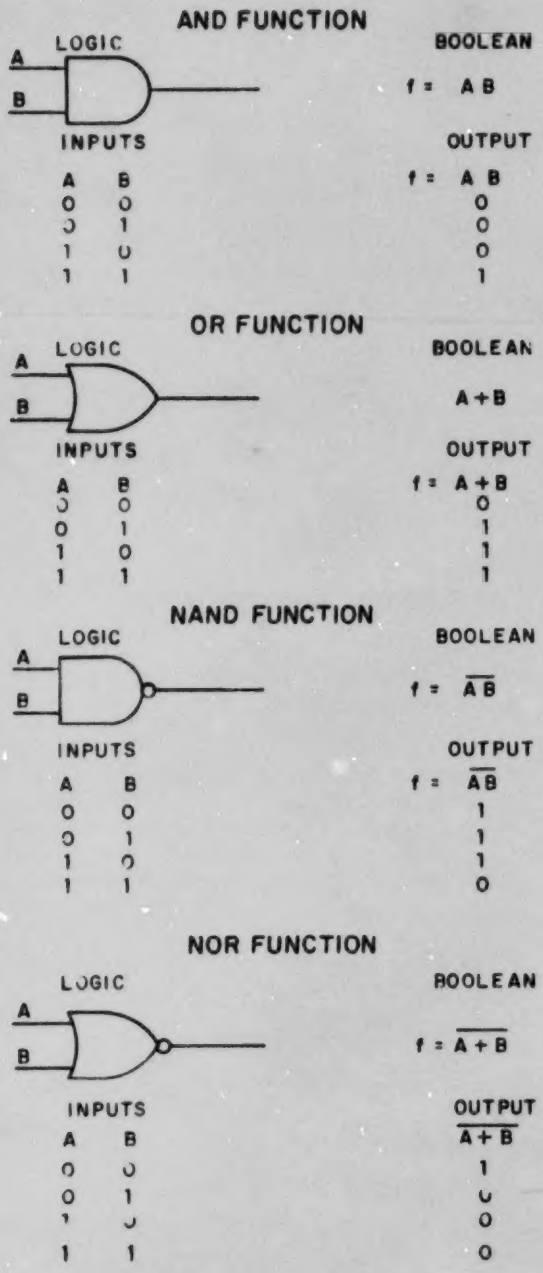


Figure 2-5.—The Four Basic Logic Circuits, with Truth Tables.

Q7, Q8, and Q9 refer to the information contained in the box below.

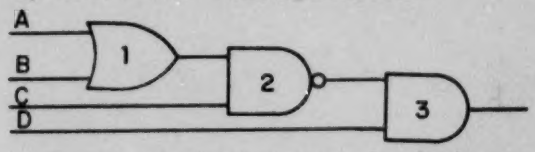
Q7. Which of the Boolean expressions shown contain NAND gates?

Q8. Which contain NOR gates?

Q9. Which contain AND and OR gates with inverted inputs?

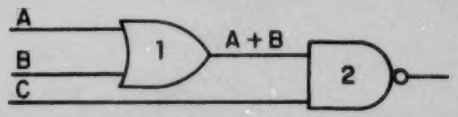
1. $AB + (C + D)\overline{EF}$
2. $(A + B) + CD + EF$
3. $(\overline{A} + B) + C\overline{D} + EF$
4. $AB + \overline{(C + D)}EF$
5. $AB + \overline{C}\overline{D}(EF)$
6. $\overline{A} + \overline{B}\overline{(CD)}\overline{(E + F)}$

From what you have learned, you should be able to draw and write the logic for any Boolean expression. As an example, write the Boolean expression for the following illustration.

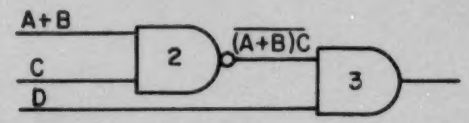


To write the Boolean expression for it, you must begin with the inputs at the left and move right, writing the output of one logic gate as the input to the next gate on its right.

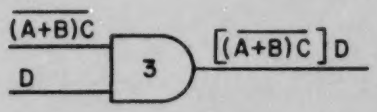
STEP 1



STEP 2

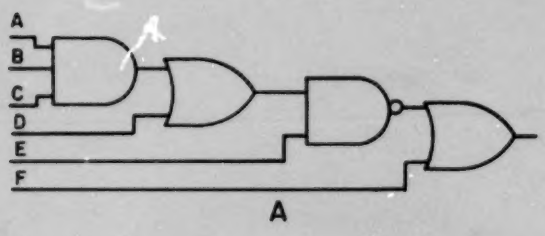


STEP 3

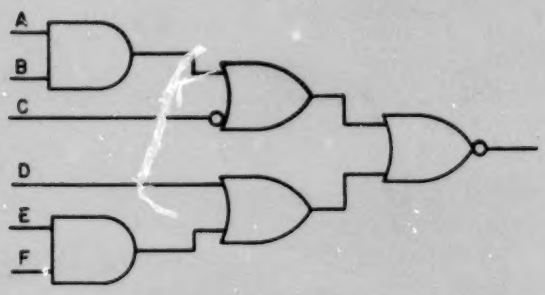


There is no problem as long as you follow the logical sequence from left to right and remember to apply the proper signs of grouping.

Q10. Write the Boolean expressions for the following logic diagrams.



A

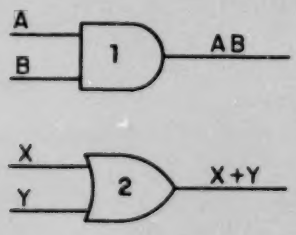


B

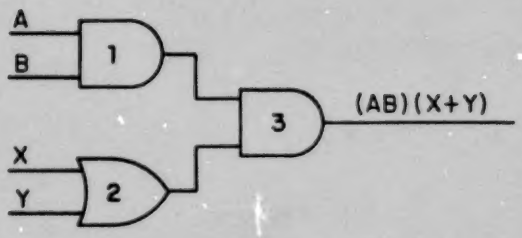
Now that you have seen how easy it is to write the Boolean expression of a logic diagram, let's turn the whole thing around and draw a logic diagram from its Boolean expression. You are given (AB) (X + Y); draw the logic diagram. First, you must separate the expression into its parts,

(AB) (X + Y)

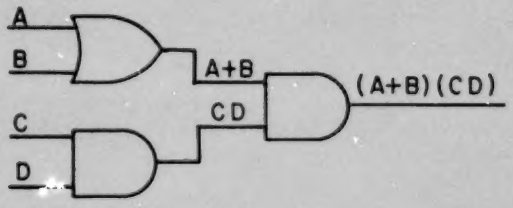
then diagram the separate expressions,



and, finally, combine them.



You may be wondering about the origin of AND gate 3. In Boolean, when two expressions are written side by side this indicates an AND function. A plus sign (+) indicates that they are joined by an OR gate. For example, (A + B) (CD) would be drawn as:

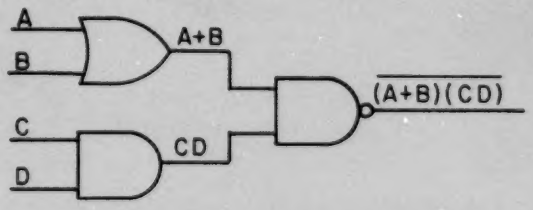


To write and expand the expression $\overline{(A + B)(CD)}$, first we place AND function dots between grouped terms, then number the terms and the grouping signs common to these terms.

Step (1) $\overline{(A + B) \cdot (C \cdot D)}$

Steps (2) (3)

In this case, the dot and solid vinculum indicate an AND gate with an inverter attached to its output. Remember, this is called a NAND gate. In addition, we also have an OR gate and an AND gate. This resultant circuit can be drawn as



A7. 1 and 6

A8. 4 and 6

A9. 3, 5 and 6

A10. A. $\overline{(ABC + D)E} + F$

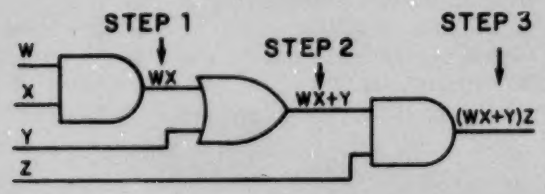
B. $\overline{(AB + \bar{C}) + (D + EF)}$

The vinculum also indicates that a common gate is used. While the preceding example may look harder than some of the previous examples, it isn't. All you have to do is separate the terms and count the gates. (Remember, the vinculum is also a term.)

The following information provides an explanation of how to find the output expression for a logic diagram.

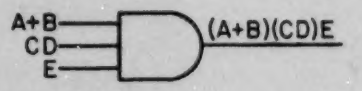
- A logic diagram is composed of two or more logic symbols.

- To find the output expression for a logic diagram, begin at the left and find the output of each logic symbol.

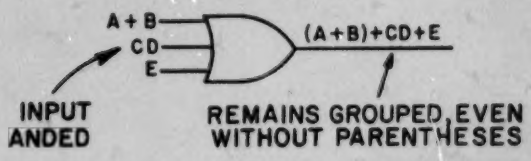


- If a logic symbol is at the extreme left of the diagram, its inputs are single letters.

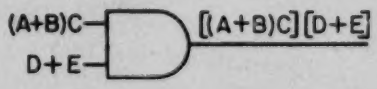
- An input signal to any symbol <u>not</u> at the extreme left may be represented by two or more letters. These letters should remain grouped in the output expression.



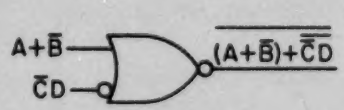
- Parentheses are used to indicate grouping, except for an ANDed input to an OR or NOR logic symbol.



- If additional grouping signs are necessary for an expression that already contains parentheses, use brackets.



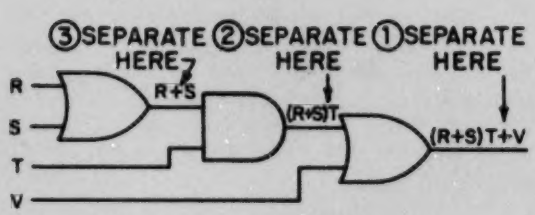
- The vinculum is used to group the portion or portions of the output expression that have been inverted. The split bar indicates individual terms inverted at the input side of the gate.



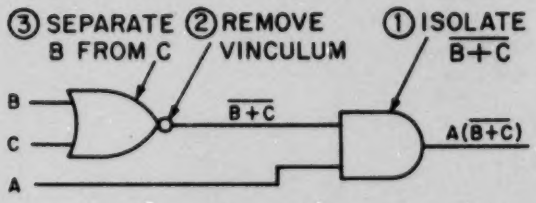
The following information provides an explanation of how to construct a logic diagram from an output expression.

- Begin drawing at the right, and work left until all inputs are single letters.

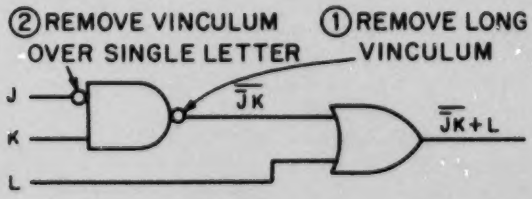
- Never separate letters within a group until that group has been separated from the rest of the expression.



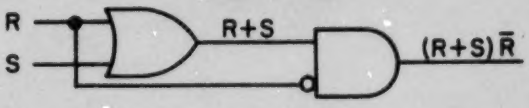
- If an expression contains a vinculum, do not remove the vinculum until you have isolated this part of the expression from the rest of the expression; and do not separate the letters under the vinculum until you have removed the vinculum.



- If a vinculum extends over more than one letter, use a NOR or NAND symbol to remove it. If a single letter is inverted, use a NOT symbol on the input.



- If a single letter is an input to more than one logic symbol, connect input lines with a dot as shown on the R input of the diagram below.



- If a circuit can accept only a certain number of signals, draw each logic symbol with only that number of inputs.

Q11. Draw the logic diagrams for the following Boolean expressions.

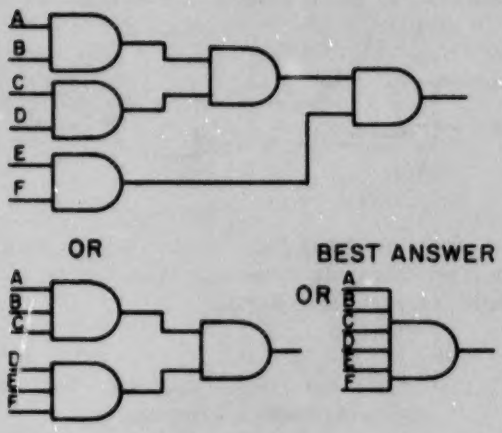
1. (AB) (CD) (EF)
2. (B + C) (D + E)
3. A + B + E (DC)
4. AB(C + D)

In problems 1, 3, and 4 you may have come up with a diagram which is different from the diagram given as the best answer. While a three-, four-, or five-gate circuit will certainly give the desired output and is correct in Boolean, the diagram that uses fewer gates to achieve the correct output is the best answer. The reason that two gates are used in problems 3 and 4 is that the OR function and the AND function are ANDed together. They can both be put into one common AND gate rather than two. This same principle can be applied to problem number one, in which the best answer is one AND gate with six inputs.

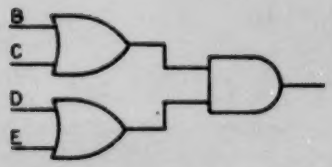
The reason problems 1, 3, and 4 were given was not to trick you, but to illustrate a point, and, incidentally, introduce you to simplification using Boolean algebra. The point to be made is that while a given logic diagram may give the desired output, the only correct one is the one that uses the fewest logic gates. The reason for this is economics; it is cheaper to build a logic function using 10 gates that it is using 100. This is where Boolean algebra shines. Once you have learned simplifications through

A11.

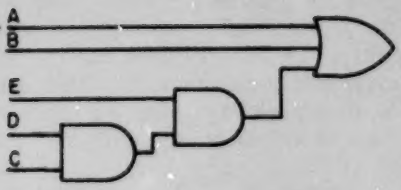
1.



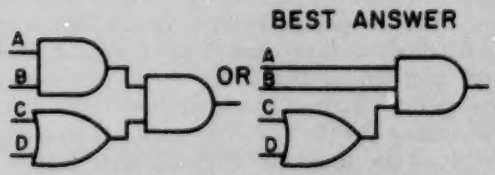
2.



3.



4.



the use of Boolean laws and theorems, you can consider yourself a master of Boolean logic.

BOOLEAN LAWS AND THEOREMS

Table 2-1 is a list of the Boolean laws and theorems used to simplify Boolean expressions and logic circuits. You do not need to memorize this table since it is a foldout which is located at the end of this topic, and is meant to be used along with the text explanation throughout the rest of this topic.

While table 2-1 may seem complicated, it really is not. Most of the laws and theorems are based on logic and observation.

Law of Identity

The Law of Identity states $A = A$, or $\overline{A} = \overline{A}$. This simply means that any expression is equal to itself. For example, if A is TRUE in one part of a Boolean expression, it is TRUE in all parts of that same expression. This is demonstrated by the following example.

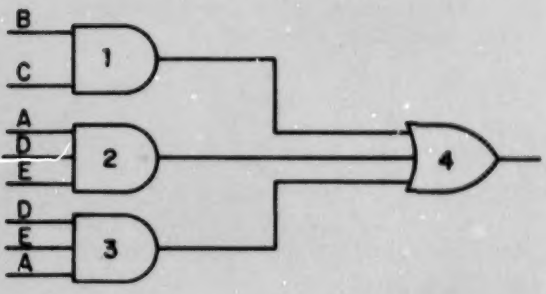
$$A + (BC) + AC$$

TRUE TRUE

Commutative Law

The Commutative Law is just as simple. It states $AB = BA$. In plain English, the equation says that when logic symbols are ANDed or ORed, the order in which they are written does not affect their value. If you are given the Boolean expression $ABC = CBA$, and all inputs (ABC) are true, the output is just as true if written as ABC or CBA or BCA. It is important to realize this. When simplifying a Boolean expression it is necessary to recognize that one part of the expression is equal to another. When these parts are the same, the output can be obtained from less circuitry. By examining the

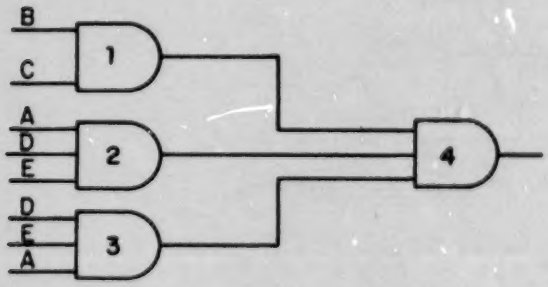
expression BC + ADE + DEA, we can see how this works. The expression can be diagramed as:



You will note that AND gates 2 and 3 have the same inputs of A, D, and E. The first law of identity states that the A of AND gate 2 is equal to the A of AND gate 3, and the same may be said for the inputs of D and E. We will get a 1, or TRUE, from AND gates 2 and 3 at the same time. Gate 4 is an OR gate requiring only one TRUE input for a TRUE output. Thus, one gate (either 2 or 3) is wasted and can be eliminated. The only time the logic circuit will give you a TRUE output is when B and C are TRUE, or ADE is TRUE. Thus, the expression BC + ADE + DEA can be written as

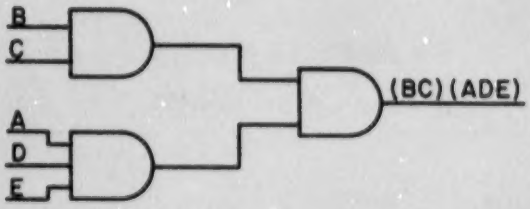
$$BC + ADE + \cancel{DEA} = BC + ADE$$

If the expression for the preceding example had been (BC) (ADE) (DEA), it could be diagramed as



You will notice that AND gates 2 and 3 will give TRUE outputs at the same time (when the same exact input conditions are met); and gate 4 will always have two of its inputs going TRUE at the same time. Thus, either gate 2 or gate 3 is extra and can be eliminated. The expression can be simplified to (BC) (ADE).

The new logic diagram will look like this



By applying the commutative law you have saved the cost of additional components.

Q12. Some of the following expressions are the same and some are not. To which expressions can the commutative law be applied?

1. $A(\overline{L} + \overline{M})$ and $(\overline{M} + L)A$
2. $A + D + BC$ and $CB + D + A$
3. $A(BC + D + \overline{E})$ and $(\overline{E} + BC + D)A$
4. $HI + L$ and $HL + I$
5. $(D + F)E$ and $F(E + D)$

Q13. Simplify the following expressions according to the commutative law.

1. $A\overline{B} + \overline{B}A + CDE + \overline{C}DE + E\overline{C}D$
2. $AB + AC + BA$
3. $C\overline{E} + \overline{A} + \overline{CE} + \overline{EC}$
4. $E\overline{B} + AG + \overline{B}E + A\overline{G}$
5. $(LMN)(AB)(CDE)(MNL)$
6. $F(K + R) + SV + W\overline{X} + VS + \overline{X}W + (R + K)F$

Associative Law

The third law in table 2-1 is the Associative Law, which states that A(BC) = ABC, or

A12. 2 and 3

A13. 1. $A\overline{B} + CDE + \overline{C}DE$

2. $AB + AC$

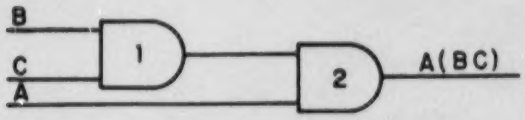
3. $\overline{CE} + C\overline{E} + \overline{A}$

4. $E\overline{B} + AG + A\overline{G}$

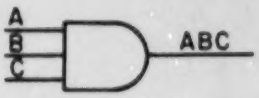
5. (LMN) (AB) (CDE)

6. $F(K + R) + SV + W\overline{X}$

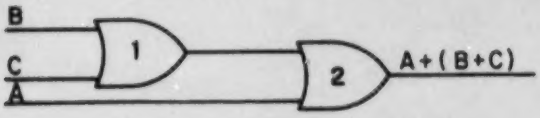
A + (B + C) is equal to A + B + C. By diagraming the expression A(BC) and A + (B + C), we can prove that the associative law is correct.



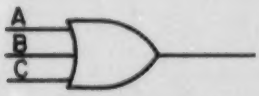
Notice that AND gate 2 in the preceding diagram requires A and the output of AND gate 1 to be TRUE. AND gate 1 requires both B and C to be TRUE for a TRUE output. The output of AND gate 2 requires that A and B and C all be TRUE for a TRUE output. Thus, A(BC) can be rewritten and rediagramed as ABC, as shown below.



The OR function of A + (B + C) can be treated the same way. First, diagram it as



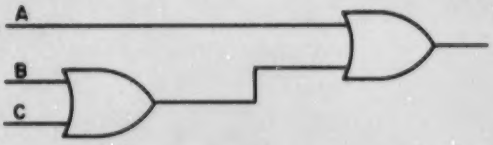
Anytime B or C is TRUE, OR gate 1 will give a TRUE output which OR gate 2 will pass as a TRUE output. When A is TRUE, OR gate 2 will also give a TRUE output. Thus, OR gate 2 will give a TRUE output when either A or B or C is TRUE. This can be diagramed as

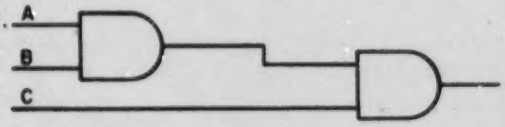


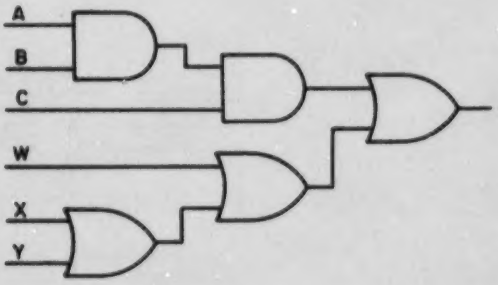
Q14. Simplify the following Boolean expressions.

1. ABC(DE)
2. BC + (DE + FG)
3. A(BC) + DC(BE)
4. W + (X + Y) + Z + (V + V)

Q15. Rediagram the following logic diagrams to their simplest forms and write their Boolean expressions.

1.


2.


3.


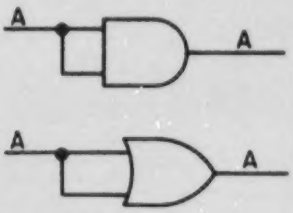
If 3 was a stumper, it's probably because you tried to work the problem from the logic diagram. While it is possible to work problems this way, it is much easier to first write the Boolean expression from the diagram, and then simplify it using the associative law. This is shown in the following example.

$$W + (X + Y) + (AB)C$$

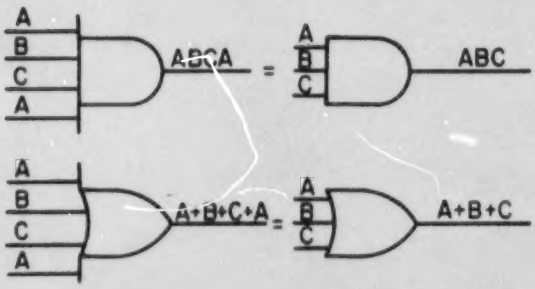
$$W + X + Y + ABC$$

Idempotent Law

The Idempotent Law is one of the easier laws of Boolean to understand and use. It can be stated as $AA = A$, or $A + A = A$, and diagramed as



Anytime A is TRUE, both inputs will be TRUE. Any term ANDed or ORed with itself will be equal to itself. This means that if A is TRUE, and it is ORed to itself, the output will be TRUE. If A is FALSE, then the output will be FALSE. While you will seldom see an expression of AA or A + A in Boolean, you might see ABCA or A + B + C + A, in which case the extra term can be eliminated as in the following illustration.



Whenever you see a Boolean expression with the same term ANDed or ORed to itself, simply remove one of the extra terms.

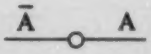
$$DED + (AB)(AB) = (DE) + (AB)$$

Q16. Simplify the following Boolean expressions. The numbers to the right of the expression indicate which of the first four Boolean laws should be used.

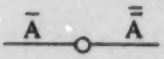
1. (RS)(RS) + A(BC) + CAB	2, 3, 4
2. $\overline{X}YZ + X + (ZY)(YZ) + X + (BC)$	2, 3, 4
3. LMLMNN	4
4. ABC + A(BC) + D + (E + F)	3, 4

Double Negative Law

Our next law is the Double Negative Law, and is written as $\overline{\overline{A}} = A$. This should be perfectly clear. If a NOT expression is brought into an inverter, it is inverted to its opposite state.



The double vinculum over the term A is just another way of showing that a NOT function has been inverted to a non-NOT function.



The bar above the $\overline{A}$ indicates the presence of an inverter. The presence of more than one vinculum above an expression is a clear giveaway that the expression is not written in its most simplified form. When simplifying, just remember this simple rule, "If there is an odd number of vincula above an expression, write it as a NOT function. If the expression has an even number of vincula, write it as a non-NOT function."

$$\overline{\overline{\overline{A}}} = \overline{A} \qquad \overline{\overline{\overline{\overline{A}}}} = A$$

A14. 1. ABCDE

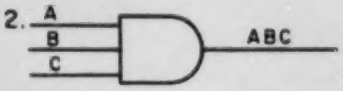
2. BC + DE + FG

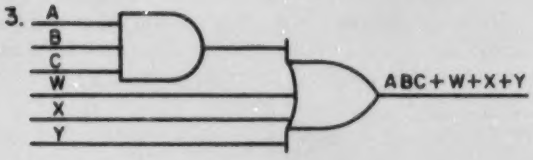
3. ABC + DCBE

4. W + X + Y + Z + V

A15.

1.

2. 

3. 

A16. 1. RS + ABC

2. $\overline{X}YZ + X + ZY + BC$

3. LMN

4. ABC + D + E + F

Q17. **Simplify the following examples.**

1. $\overline{\overline{AB}} + \overline{\overline{X}}$

2. $\overline{\overline{DE}}$

3. $\overline{\overline{(\overline{R} + S)}\overline{T}}$

Question number three may appear to be a little tricky, but it isn't. When removing vincula, just remember the vinculum above each separate term is counted for that term. Therefore, if we have $\overline{\overline{(\overline{R} + S)}\overline{T}}$ you will notice that R has three vincula over it, while S has only two. The rule states that an even number of vincula results in a non-NOT function, and an uneven number of vincula results in a NOT function. Therefore,

$$\overline{\overline{(\overline{R} + S)}\overline{T}} = (\overline{R} + S)\overline{T}$$

Q18. **Simplify the following expressions using the double negative law and other laws you have learned.**

1. $\overline{\overline{C}} + \overline{DF} + \overline{\overline{FD}} + G + \overline{\overline{C}}$

2. $\overline{\overline{(Q + R} + \overline{S})} + (R + Q)$

3. $(M\overline{\overline{NP}} + Q)L + \overline{\overline{K}}$

4. $\overline{\overline{WXY}} + Z)\,(Z + \overline{\overline{XWY}})$

Complementary Law

Law number six is the Complementary Law. It is stated $A\overline{A}$ = FALSE or 0, or $A + \overline{A}$ = TRUE or 1.

In the first case of $A\overline{A}$, we have an AND gate with both states of A as inputs. Therefore, one input is TRUE (1) and one is FALSE (0).

AND gates require each input to be TRUE (1) to get a TRUE (1) output. Therefore, the only possible output is FALSE (0). This is shown in figure 2-6.

The same applies to the OR function of $A + \overline{A}$ = TRUE (1). The OR gate requires that only one input be TRUE (1) to get a TRUE (1) output. Therefore, with inputs of $A + \overline{A}$, the

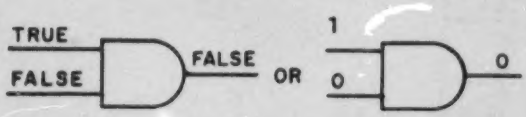


Figure 2-6.—$A\overline{A}$ = FALSE (0).

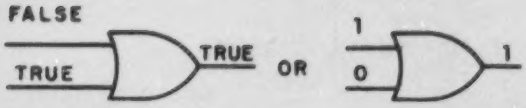
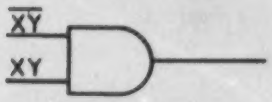
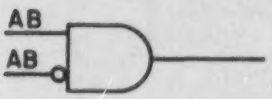


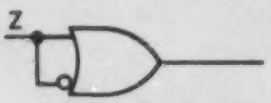
Figure 2-7.—$A + \overline{A}$ = TRUE (1).

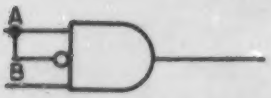
term that represents a TRUE (1) (either A or $\overline{A}$) will cause the OR gate to output a TRUE (1). This is shown in figure 2-7.

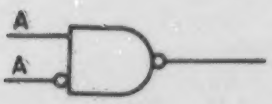
Q19. **In the following logic gates, you will find some that will have TRUE outputs and some that will have FALSE outputs. Indicate which outputs are TRUE by marking each with a 1, and those that are FALSE mark with a 0.**

1. 

2. 

3. 

4. 

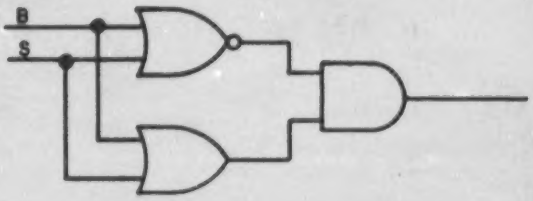
5. 

6.

Q20. **What is the logic state of**

$$\overline{A} + \overline{CD} + DC + A$$

Q21. **Using a 0 to represent a FALSE state and a 1 to represent a TRUE state, what is the output of the logic diagram shown below?**



You have now learned the first six laws of Boolean. Use them to simplify the following expressions. HINT: Whenever an expression can be simplified down to the point where only the complementary law applies, then give your answer as either a 1 (for TRUE) or a 0 (for FALSE).

Example:

$$(A + B)(A + B)(\overline{A + B})$$

In this example you first apply the law of identity.

$$(A + B) = (A + B)$$

Since this is true, you can now apply the idempotent law.

$$(A + B)(A + B) \rightarrow (A + B)(\cancel{A + B}) = (A + B)$$

This leaves you with $(A + B)(\overline{A + B})$ and, as $(\overline{A + B})$ is the complement of $(A + B)$, you can apply the complementary law as $(A + B)(\overline{A + B}) = 0$.

A17. 1. $\overline{\overline{AB} + X}$

2. $\overline{\overline{DE}}$

3. $(\overline{\overline{R} + S})\overline{T}$

A18. 1. $C + \overline{DF} + G$

2. $Q + R + S$

3. $(MNP + Q)L + \overline{K}$

4. $WXY + Z$

A19. 1. 0

2. 0

3. 1

4. 0

5. 1

6. 0

A20. True or 1

A21. 0

Q22. $X\overline{X}\overline{\overline{X}}$

Q23. $(K + \overline{\overline{K}})K\overline{K}$

Q24. $\overline{B\overline{AC}} + A(\overline{C}B)$

Q25. $\overline{L + \overline{MP}}$ $\overline{\overline{\overline{(MP)} + L}}$

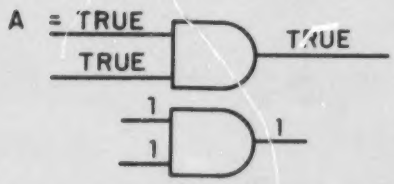
Q26. $\overline{W(XY)Z} + (WX)\overline{\overline{YZ}}$

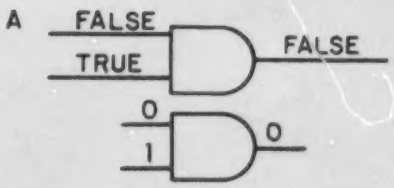
Q27. $(DEA + B)\overline{(D + EBA)}$

Q28. $\overline{\overline{\overline{Q} + \overline{R}\overline{S}\overline{T}}}$ $\overline{\overline{\overline{(\overline{ST})}\overline{R} + Q}}$

Law of Intersection

The Law of Intersection is a variation of the complementary law. The law of intersection states A · TRUE (1) = A, or A · FALSE (0) = FALSE (0). The truth of this law is obvious. In the first case (A · TRUE (1) = A), if one input to an AND gate is already TRUE (1), then the output will depend on the state of A only. If A is TRUE (1), the AND gate will have two TRUE (1) inputs and the output will be TRUE (1). If A is FALSE (0), the AND gate will have one FALSE (0) input (in this case A) and one TRUE (1) input; therefore, the output will be FALSE (0). In both cases, the output is the same state as A. This is shown in the following diagram.

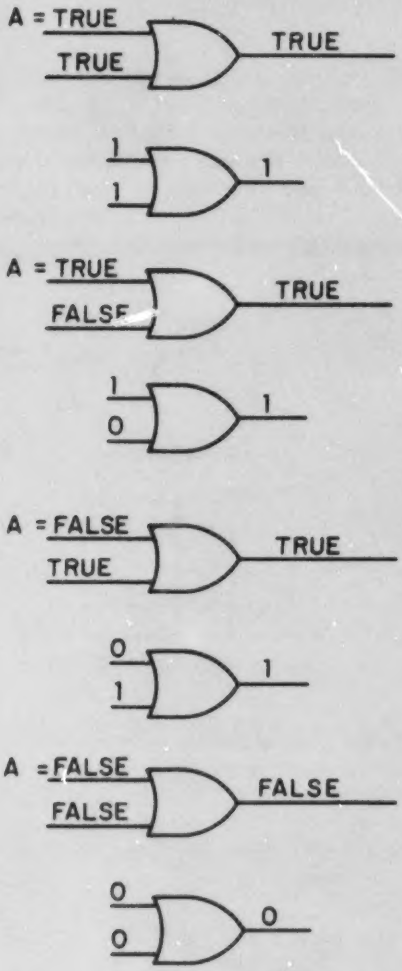




Law of Union

The Law of Union is the same as the law of intersection, only it is applied to OR gates. It states A + TRUE(1) = TRUE (1), or A + FALSE (0) = A. In the former statement, A + TRUE, the OR gate has one input labeled TRUE (1); its output, therefore, will be TRUE (1) regardless

of the state of A. In the statement A + FALSE (0) = A, we have just the opposite case. One input is labeled FALSE (0) and the only way we can get a TRUE (1) output from the OR gate is if A is TRUE (1). Therefore, only the state of A will determine the output state of the OR gate. These conditions are shown in the following logical diagrams.



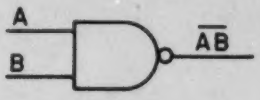
As you can see from the preceding example, when A is ORed with a TRUE (1), the output will always be TRUE (1), regardless of the state of A. Likewise, when A is ORed with a FALSE (0), the output will depend on the state of A only.

Q29. Use the laws of intersection and union to simplify the following expressions.

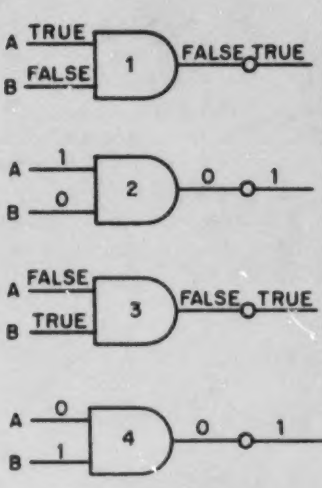
1. $TRUE \cdot \overline{B}$
2. $1 + A$
3. $0 \cdot A$
4. $AB \cdot 0$
5. $(\overline{AC}) \cdot TRUE$

DeMorgan's Theorem

DeMorgan's theorem is concerned with NAND and NOR logic gates. The first part of the law states $\overline{AB} = \overline{A} + \overline{B}$. The solid vinculum indicates the presence of the NAND gate, as shown in the following diagram.



You will notice, both from the diagram and from the discussion of NAND gates, that the only time we can get a TRUE, or 1, output from a NAND gate is if one of the inputs is FALSE, or 0.



A	B	OUTPUT
0	0	1
0	1	1
1	0	1
1	1	0

NAND GATE TRUTH TABLE

A22. 0

A23. 0

A24. 1

A25. $\overline{L + MP}$

A26. 1

A27. $(DEA + B)\,\overline{(D + EBA)}$

A28. 0

A29. 1. $\overline{B}$

2. 1

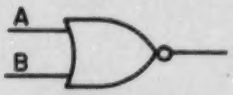
3. 0

4. 0

5. $\overline{AC}$

In each of the preceding NAND gates, it is the FALSE or 0 input that causes the TRUE or 1 output. In NAND gates 1 and 2 we get a TRUE (1) output when B is FALSE or when $\overline{B}$ is TRUE. (Remember, $\overline{B}$ is B complemented.) NAND gates 3 and 4 output a TRUE (1) output when $\overline{A}$ is TRUE (A is FALSE). Therefore, anytime that $\overline{A}$ or $\overline{B}$ is true, we would get a TRUE, or 1, from the illustrated NAND gates; or, as the law states, $\overline{AB} = \overline{A} + \overline{B}$.

The second half of DeMorgan's theorem is stated as $\overline{A + B} = \overline{A}\,\overline{B}$. The term $\overline{A + B}$ is obviously the output from a NOR circuit and can be diagramed as



The only time we can get a TRUE from a NOR circuit is when both inputs are FALSE. When A and B are both FALSE, $\overline{A}$ and $\overline{B}$ are both TRUE. We can therefore state that $\overline{A + B} = \overline{A}\,\overline{B}$, and that both of these expressions will equal a TRUE at the same time. The following diagram illustrates this point.

A	B	OUTPUT
0	0	1
0	1	0
1	0	0
1	1	0

NOR GATE TRUTH TABLE

DeMorgan's theorem is one of the most useful Boolean tools that you can master. Through it you can split or join vincula to greatly simplify most Boolean expressions. The expression $RS + TV + \overline{R} + \overline{S}$ can be simplified to its basic terms by the use of DeMorgan's theorem.

First Step: Pull out the Boolean Law Summary foldout sheet.

Second Step: Apply DeMorgan's theorem, $RS + TV + \overline{RS}$.

Third Step: Apply the commutative law, $RS + \overline{RS} + TV$.

Fourth Step. Apply the complementary law, $1 + TV$.

Fifth Step: Apply the law of union, $1 + TV = 1$.

Our answer, then, is 1 or TRUE. Simple, isn't it? Now, it's your turn. Apply DeMorgan's theorem to the following problems.

Q30. 1. $\overline{V + A + L}$

2. $\bar{A} + \bar{B} + \bar{C} + \bar{D}$

3. $\overline{WXYZ}$

4. $\overline{ABC} + D$

In problem 4, the vinculum covers only part of the expression. In a case like this, SIGNS and TERMS under the vinculum change state. Signs outside the vinculum do not. Also, remember that all signs under the vinculum change. If you had trouble with these problems, use the following procedure in simplifying $\overline{WX + Y}$ to $(\bar{W} + \bar{X})(\bar{Y})$. It is done in the following way. First, break the vinculum at all AND and OR signs and then change all signs under the breaks. However, when you do this, it is recommended that it be done in single steps and that the logic equation be rewritten after each step. For example, the logic expression given above would have been simplified in the following manner.

Original expression	$\overline{WX + Y}$
Working from right to left, break the vinculum above the first sign	$\overline{WX} + \bar{Y}$
Change the sign and rewrite	$\overline{WX}(\bar{Y})$
Break the vinculum over the next sign	$\bar{W}\bar{X}(\bar{Y})$
Change the sign and rewrite	$(\bar{W} + \bar{X})(\bar{Y})$

Now that you have seen how it is done, try simplifying the following expressions using DeMorgan's Theorem.

Q31. 1. $\overline{A + BD}$

2. $\bar{H}(\bar{J} + \bar{L})$

3. $(\bar{T} + \bar{V})\bar{W}$

4. $\overline{R + LM} + X$

5. $\overline{(TS + R)Q}$

6. $\overline{AB}(\bar{Y} + \bar{Z})$

7. $\bar{F}(\bar{G} + \bar{V}) + \bar{H}$

8. $D + \overline{G(E + F) + H}$

9. $A\overline{BC} + \overline{Z + T}$

10. $\overline{(QR + S)N} + \overline{PM}$

So far, all the problems you have handled here have had single vinculum over them; but, you might be wondering how to simplify expressions with more than one vinculum over them? For example

$$\overline{\bar{B} + CD}$$

First, split the long vinculum and change the signs (using the single-step operation shown above) until the vinculum is completely split and all signs have been changed.

$$\bar{\bar{B}}(\bar{C} + \bar{D})$$

Then, apply the double negative law.

$$B(\bar{C} + \bar{D})$$

Q32. Simplify the following expressions.

1. $\overline{(A + B)(\overline{L + M})}$

2. $\overline{WX\bar{Y}} + \overline{A\bar{B}}$

A30.

1. $\overline{V}\overline{A}\overline{L}$
2. $\overline{ABCD}$
3. $\overline{W} + \overline{X} + \overline{Y} + \overline{Z}$
4. $\overline{A} + \overline{B} + \overline{C} + D$

A31.

1. $\overline{A}(\overline{B} + \overline{D})$
2. $\overline{H + JL}$
3. $\overline{TV + W}$
4. $\overline{R}(\overline{L} + \overline{M}) + X$
5. $(\overline{T} + \overline{S})\overline{R} + \overline{Q}$
6. $\overline{AB + YZ}$ or $\overline{A} + \overline{B}(\overline{Y} + \overline{Z})$
7. $\overline{(F + GV)H}$
8. $D + \overline{G} + (\overline{E}\overline{F})\overline{H}$
9. $A(\overline{B} + \overline{C}) + (\overline{Z}\overline{T})$
10. $(\overline{Q} + \overline{R})\overline{S} + \overline{N} + \overline{P} + \overline{M}$

A32.

1. $\overline{A}\overline{B} + L + M$
2. $(\overline{W} + \overline{X} + Y) + (\overline{A} + B)$

DeMorgan's theorem allows us to not only combine and split vincula, but, when used with the double negative law, to add vincula to expressions. By using both methods, it is sometimes very easy to simplify an expression. If you are given the expression

$$[B + \overline{C(D + E)}]\,[\overline{B}C(D + E)] + A$$

by splitting the vinculum inside the first brackets you would get

$$[B + \overline{C} + \overline{D}\overline{E}]\,[\overline{B}C(D + E)] + A$$

This does not simplify the expression at all.

The expression can be simplified by applying DeMorgan's theorem and adding two vincula over B. This is allowable because the double negative law states that there is no difference between B and $\overline{\overline{B}}$. Going back to the original expression we now have

$$[\overline{\overline{B}} + \overline{C(D + E)}]\,[\overline{B}C(D + E)] + A$$

Now, join the vincula in the first part of the expression to get

$$\overline{[\overline{B}C(D + E)]\,[\overline{B}C(D + E)]} + A$$

You should note that the expressions

$$[\overline{B}C(D + E)]\,[\overline{B}C(D + E)]$$

are complements of each other and are ANDed together. The complementary law applies to

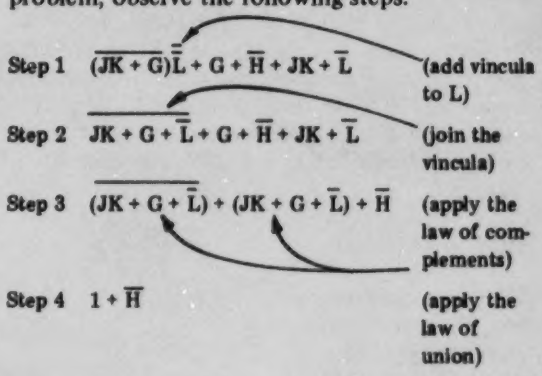
$$A\overline{A} = 0 \text{ or } \overline{[\overline{B}C(D + E)]\,[\overline{B}C(D + E)]} = 0$$

What we then have,left is 0 + A. By applying the law of union to A + 0 = A we get the simplified expression A.

This means that any time that A is TRUE, the logic circuits expressed by $[B + \overline{C(D + E)}]\,[\overline{B}C(D + E)]\,A$, would give a TRUE output. "A" can, therefore, replace the entire logic circuit.

Q33. Simplify $\overline{(JK + G)}L + G + \overline{H} + JK + \overline{L}$.

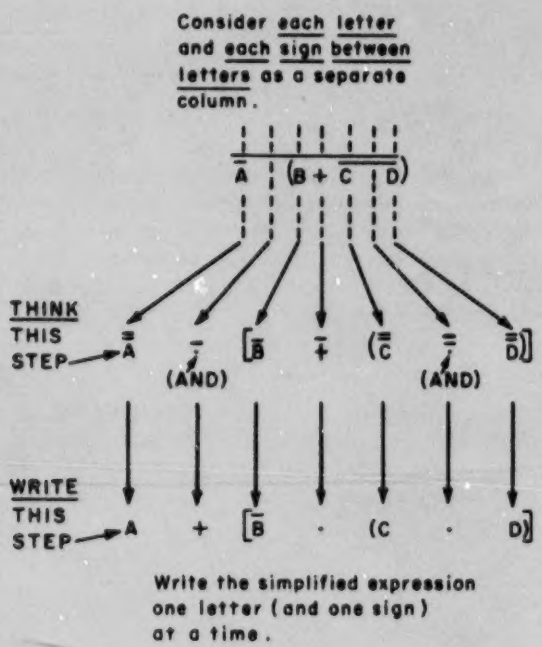
If you had difficulty with the preceding problem, observe the following steps.


Step 1 $\overline{(JK + G)}\overline{\overline{L}} + G + \overline{H} + JK + \overline{L}$
(add vincula to L)
Step 2 $\overline{JK + G + \overline{L} + G + \overline{H} + JK + \overline{L}}$
(join the vincula)
Step 3 $\overline{(JK + G + \overline{L}) + (JK + G + \overline{L}) + \overline{H}}$
(apply the law of complements)
Step 4 $1 + \overline{H}$
(apply the law of union)
Step 5 1 or TRUE

You have been shown the conventional way to simplify an expression with more than one vinculum over the same letter. For an expression like $\overline{(A + B)C} + \overline{(D + E)F}$, this method would take many steps. A quicker, easier method follows.

If an odd number of vincula are removed, the sign changes from AND to OR, or from OR to AND. If an even number of vincula are removed, the sign will not change. This quick method is illustrated next.

Simplify $\overline{\bar{A}(B + \overline{CD})}$.


Consider each letter and each sign between letters as a separate column.
THINK THIS STEP
WRITE THIS STEP
(AND)
(AND)
Write the simplified expression one letter (and one sign) at a time.

$A + [\bar{B}(CD)]$ may be simplified to $A + \bar{B}CD$.

Simplify the following expressions, using the new method. Be careful to use grouping signs in the right places.

Q34. $\bar{Q} + \overline{R(\bar{S} + T)}$.

Q35. $\overline{D\bar{E}(F + \overline{GH}) + J}$

Q36. $\overline{(\bar{W}\overline{XY} + Z)}\overline{TV}$

Q37. $\overline{\overline{(\bar{K} + L)}M + N(\bar{H} + J)}$

Unless you can manipulate an expression so that you have a group of letters and its complement (so that both equal 0, or both equal 1), the best procedure for simplifying is this:

First split the vincula according to DeMorgan's theorem. (Remember to group letters as they were originally grouped).

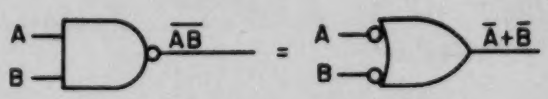
Then apply other laws, in whatever order seems most appropriate.

Q38. Simplify the following expressions.

1. $\overline{(E + \bar{E})(\overline{FG} + \bar{G} + \bar{F})}$
2. $\overline{\overline{(KL + M)}(\bar{K} + \bar{L})\bar{M}}$
3. $[A + \bar{B}C + \bar{A}(B + \bar{C})]D$
4. $\overline{CG\bar{T}(\overline{KF} + \bar{T})}$
5. $[\bar{W}\bar{X} + Y][\bar{Y}(W + X)] + Z$
6. $\overline{(\overline{Q + R})\,\overline{(S + \bar{T})}\bar{Q}\bar{R}\bar{S}}$

Q39. To see how many logic symbols (including inverters) that may be eliminated by the laws you have learned, diagram problem 6 located in question 38.

Q40. According to DeMorgan's theorem, $\overline{AB} = \bar{A} + \bar{B}$ so



Inputs to both symbols are identical, and the outputs are equal. Therefore, an OR gate with inverted inputs could be substituted for a/an ____ gate, and vice versa.

A33. "1" or TRUE

A34. $\overline{Q} + \overline{R} + S + T$

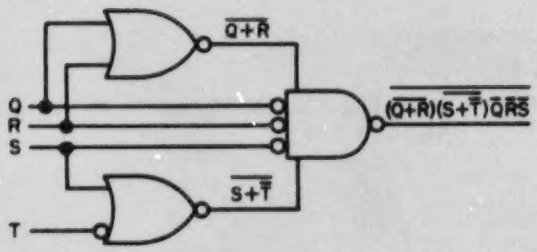
A35. $(\overline{D} + E + \overline{F}GH)\overline{J}$
HINT: The first step you wrote down should have been $\overline{D\overline{E}(F + \overline{GH})} \cdot \overline{J}$

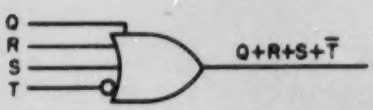
A36. $(W + XY)(\overline{T} + \overline{V})\overline{Z}$
HINT: First step
$[(W + X \cdot Y) \cdot \overline{Z}] \cdot [\overline{T} + \overline{V}]$

A37. $\overline{R}\overline{L}MN(\overline{H} + J)$

A38.
1. FG (HINT: split vincula, then simplify)
2. KL + M (HINT: split vincula, then simplify)
3. D (HINT: join vincula to form complement, then simplify)
4. 1 (HINT: split vincula, then simplify)
5. Z (HINT: join vincula to form complement, then simplify)
6. $Q + R + S + \overline{T}$

A39.





A40. NAND

Following is a summary of the principles and applications of DeMorgan's Theorem.

1. DeMorgan's theorem is used to split or join vincula.

$$\overline{AB} = \overline{A} + \overline{B}$$

$$\overline{A + B} = \overline{A}\,\overline{B}$$

2. When a vinculum is split, or when several vincula are joined to form one long vinculum, every sign over which the splitting or joining takes place changes from OR to AND, or from AND to OR. If vincula are joined, these signs change: $AB + CB + DC$

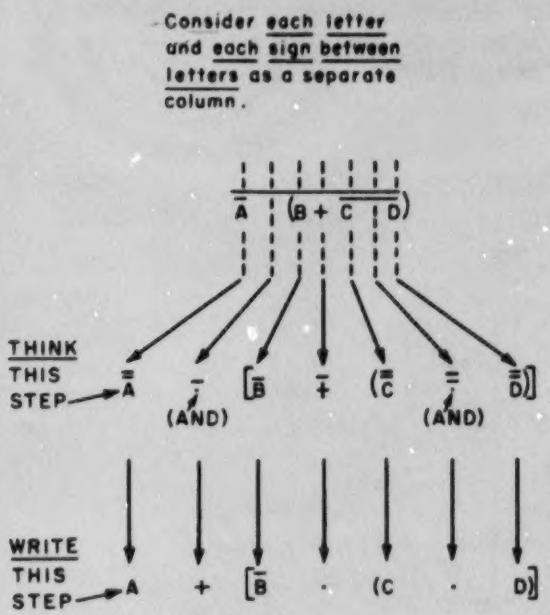
3. Whenever an even number of vincula (two, for example) are joined or split, the sign stays the same. Whenever an odd number (three, for example) are joined or split, the sign changes.

4. When you change signs, preserve the original grouping.

$\overline{A + BC}$ becomes $\overline{A}(\overline{B} + \overline{C})$

5. An expression is in its simplest form only if no letter has more than one vinculum over it.

6. Quick method for splitting vincula:


Consider each letter and each sign between letters as a separate column.
THINK THIS STEP
WRITE THIS STEP
(AND)
(AND)
Write the simplified expression one letter (and one sign) at a time.

$A + [\overline{B}(CD)]$ may be simplified to $A + \overline{B}CD$.

7. Two ways to simplify an expression by applying DeMorgan's theorem are:

a. Join vincula to form the complement of a term or expression:

$AB + C + \overline{A} + \overline{B}$

$AB + C + \overline{AB}$

$1 + C$

1

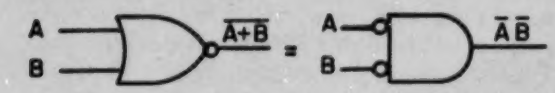
b. Split all vincula, then apply the other laws for further simplification:

$\overline{\overline{(AB + \overline{C})DC}}$

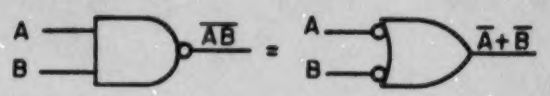
$AB + \overline{C} + \overline{D} + \overline{C}$

$AB + \overline{C} + \overline{D}$

8. NOR gate = AND gate with inverted inputs.



NAND gate = OR gate with inverted inputs.

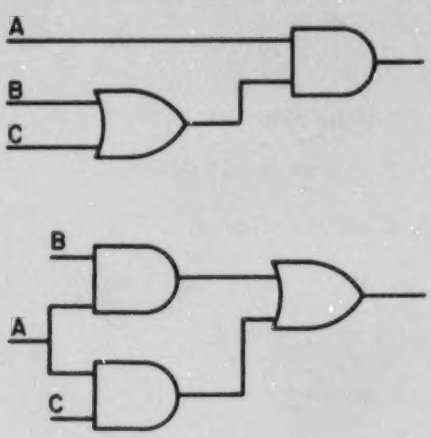


The Distributive Law

The last three laws on your list (table 2-1) are used mainly to manipulate Boolean expressions so that the other laws may be applied to simplify the expression.

The Distributive Law states that $A(B + C) = AB + AC$. For those of you familiar with normal algebra, this is nothing more than multiplication of terms; but, as this is a course in Boolean algebra, all laws must be capable of being proved using logic gates. We will now prove the truth of the distributive law.

To get a 1 or TRUE output from $A(B + C)$, A must be TRUE or 1 and either B or C must be 1 or TRUE. We now have two possible combinations of logic that will give us a 1 or TRUE output at the same time as $A(B + C)$. These are, $AB + AC$, as shown below.



While this may seem to violate the purpose of simplification using Boolean algebra, it really doesn't; because, as we stated above, the main purpose of the distributive law is manipulation. For example, if you are given $W(X + Y) + WY$, you can see that, without manipulation, the expression can not be simplified. By applying the distributive law, we get $WX + WY + WY$; then, by applying the idempotent law, we get $WX + WY$.

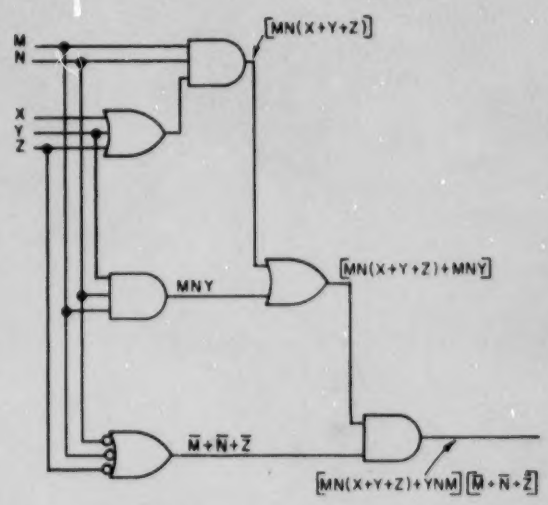
Q41. Apply the distributive law to the following expressions.

1. $D(E + F + G)$
2. $A(A + B + D)$
3. $V(W + Y + XZ)$
4. $JK(L + MN)$
5. $\overline{H}J + JK\overline{L} + GJM$
6. $LMNP + LQR$

Q42. Now, use the distributive law in conjunction with the laws already covered to simplify the following expressions.

1. $A(WX + AB + A) + AWX$
2. $ZW(ABWZ) + A + (A + B)$
3. $X(\overline{X}B) + A\overline{(\overline{A}C + B)}$
4. $(TV)(T + W)(U + V)(V + W)(\overline{TV}) + Y$
5. $(BC + GB + JG)BJ$
6. $AC + AD + A\overline{C} + (B + D)(\overline{B}\overline{D})$
7. $[MN(X + Y + Z) + YNM][\overline{M} + \overline{N} + \overline{Z}]$

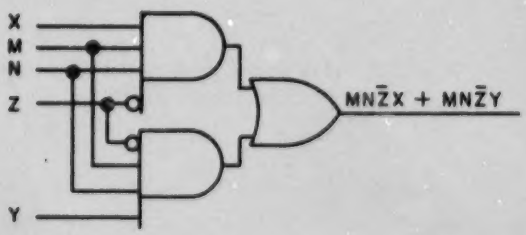
Now that you have finished the problems, use the distributive law to prove the usefulness of Boolean algebra for logic simplification. Below is the logic diagram for the expression used in problem 7.



HINT:

1. Apply the distributive law of $A(B + C) = AB + AC$.
2. Apply DeMorgan's theorem of $\overline{A} + \overline{B} = \overline{AB}$ and the distributive law.

While the preceding logic diagram is impressive, it is also expensive to build. The following simplified one is cheaper and will fulfill all the functions of the preceding one. (It is also the diagram of answer 7. If you didn't get the same answer, check it against the example given after the logic diagram.)



Given: $[MN(X + Y + Z) + YNM]\ [\overline{M} + \overline{N} + \overline{Z}]$

Step 1

$(MNX + MNY + MNZ + YNM)\ (\overline{M} + \overline{N} + \overline{Z})$ — distributive and idempotent laws

Step 2

$(MNX + MNY + MNZ)\ (\overline{MNZ})$ — DeMorgan's theorem

Step 3

$\overline{MNZ}MNX + \overline{MNZ}MNY + (\overline{MNZ}MNZ) = \overline{MNZ}MNX + \overline{MNZ}MNY$ — distributive and complementary laws

Step 4

$\overline{MNZ}MN\ (X + Y)$ — distributive law

Step 5

$MN(\overline{MN} + \overline{Z})\ (X + Y)$ — distributive law and DeMorgan's theorem

Step 6

$(MN\overline{MN} + MN\overline{Z})\ (X + Y)$ — distributive law and complementary law

Step 7

$MN\overline{Z}\ (X + Y)$

Step 8

$MN\overline{Z}X + MN\overline{Z}Y$

Now that you have seen how it is done, and worked a few problems, go back and rework any problem you may have missed.

The Law of Absorption

The next law on your pullout is the Law of Absorption. In brief, it states $A(A + B) = A$; $A + AB = A$.

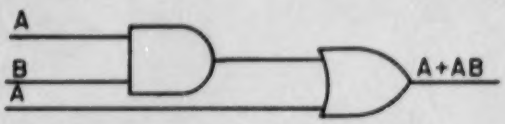
The law of absorption is easy enough to prove. First, apply the distributive law to $A(A + B)$.

$$AA + AB$$

Then, the idempotent law

$$AA + AB = A + AB$$

Then, draw out the logic diagram.



Anytime that A is equal to 1, we will get a 1 out of the OR gate. Anytime A is equal to 0, we will not get a 1 from either the AND or the OR gate. Therefore, anytime A is equal to 1 from either the AND or OR gate, we will get a 1 output regardless of the state of B.

Now that we have proven that the law of absorption is valid, let's find out how to use it for simplification.

Apply the law of distribution to the expression $A + AB = A(1 + B)$. You may wonder how we came up with $1 + B$. This is an application of the law of intersection which states that $A = A \cdot 1$. Therefore, the original expression could have been written as $A \cdot 1 + AB$; in which case, it would mean that both 1 and B were ANDed together with A. If $A(1 + B)$ is correct, we can next apply the law of union which states that $A + 1 = 1$; therefore, the expression $1 + B$ is equal to 1 and what we have is $A \cdot 1 = A$.

Taken in steps, the preceding example would look like this:

Step 1	$A + AB$	
Step 2	$A \cdot 1 + AB$	law of intersection
Step 3	$A(1 + B)$	law of distribution
Step 4	$A(1)$	law of union
Step 5	$A \cdot 1 = A$	law of intersection

A41. 1. $DE + DF + DG$

2. $AA + AB + AD$

3. $VW + VY + VXZ$

4. $JKL + JKMN$

5. $J(\overline{H} + K\overline{L} + GM)$

6. $L(MNP + QR)$

A42. 1. $AWX + AB + A = A$

2. $ZWAB + A + B$

3. $\overline{A}\overline{B}$

4. Y

5. $BJC + BJG$

6. $\overline{A}$

7. $MN\overline{Z}X + MN\overline{Z}Y$

Let's try another one. Given $VW + W + WX$, simplify it according to the law of absorption.

1. First, factor out the common term according to the distributive law.

$$W(V + 1 + X)$$

2. Check to see if the law of union applies. If it does, rewrite the problem as:

$$W(V + 1 + X) \rightarrow W(1)$$

3. Apply the law of intersection.

$$W \cdot 1 \rightarrow W$$

The law of absorption can also be used to simplify a different form of expression. For example, if you are given $A(B + C + A)$, first apply the distributive law.

$$A(B + C + A) \rightarrow AB + AC + AA$$

Next, apply the idempotent law.

$$AB + AC + AA \rightarrow AB + AC + A$$

Reapply the distributive law to factor out the common term.

$$AB + AC + A \rightarrow A(B + C + 1)$$

Apply the law of union.

$$A(B + C + 1) \rightarrow A \cdot 1$$

Then, apply the law of intersection.

$$A \cdot 1 = A$$

As you can see, the law of absorption can be applied as a multiple step operation. But if you are short of paper, it is easier to just apply the law as

$$A(A + B) = A; A + AB = A$$

Q43. **Simplify the following expressions.**

1. $R(S + T + R)$

2. $(AB + DE + V)DE$

3. $(XY + WZ + X)X$

4. $K + KL + KM$

5. $VW + W + WX$

The last five practice problems probably did not present you with too much of a challenge.

The problems were straightforward applications of the absorption law. As you are aware of by now, things are seldom simple in Boolean. For example, if you are given $A + BC + ABC$, you might, at first glance, believe the absorption law does not apply because we have no common character in the term. The absorption law can be applied in the following manner.

Step 1 Apply the commutative law.

$$A + BC + ABC \rightarrow A + ABC + BC$$

Step 2 Apply the associative law.

$$A + ABC + BC \rightarrow (A + ABC) + BC$$

Step 3 Apply the distributive law.

$$(A + ABC) + BC \rightarrow A(1 + BC) + BC$$

Step 4 Apply the law of union.

$$A(1 + BC) + BC \rightarrow A \cdot 1 + BC$$

Step 5 Apply the law of intersection.

$$A1 + BC \rightarrow A + BC$$

At this point it should be noted that most laws of Boolean are applied both backward and forward. This means that you can replace the equal signs with double arrows. For example:

$$A + (B + C) = A + B + C$$

or

$$A + (B + C) \overleftrightarrow{} A + B + C$$

Whenever you simplify, look at all expressions from both directions of the laws given in table 2-1.

Q44. With this fact in mind, use laws that have already been covered to simplify the following expressions.

1. $ST + VW + RST$
2. $TUV + XY + Y$
3. $F(E + F + G)$
4. $(PQ + R + ST)TS$
5. $ABC + CB$
6. $\overline{\overline{D}}\,\overline{DE}$
7. $Y(W + X + \overline{\overline{Y} + \overline{Z}})Z$
8. $JKL + J$
9. $(BE + C + F)C$
10. $MNP + QR + \overline{\overline{M} + \overline{N}}$

The Law of Common Identities

The last law used for Boolean simplification is the Law of Common Identities. It is nothing more than an application of the distributive, complementary, union, and intersection laws. The law states $A(\bar{A} + B) = AB$, or $A + \bar{A}B = A + B$. Take the first part, $A(\bar{A} + B) = AB$, and apply the following laws.

Step 1 The distributive law

$$A(\bar{A} + B) \rightarrow \bar{A}A + AB$$

Step 2 The complementary law

$$A\bar{A} + AB \rightarrow 0 + AB$$

Step 3 The law of union

$$0 + AB \rightarrow AB$$

Thus, we have the proof for the law of common identities of

$$A(\bar{A} + B) = AB$$

A43.

1. R
2. DE
3. X
4. K
5. W

A44.

1. ST + VW
2. TUV + Y
3. F
4. ST
5. BC
6. D
7. YZ
8. J
9. C
10. MN + QR

The common identities law can be proven using the distributive law, (A + BC) = (A + B)(A + C). Look at the common identities law.

$$A + \overline{A}B = A + B$$

Step 1 Apply the distributive law.

$A + \overline{A}B \rightarrow (A + \overline{A})(A + B)$

Step 2 Apply the complementary law.

$(A + \overline{A})(A + B) \rightarrow 1(A + B)$

Step 3 Apply the law of intersection.

$1(A + B) \rightarrow A + B$

Expressions like $A(\overline{A} + B)$ and $A + \overline{A}B$ occur frequently in Boolean algebra. If you learn to recognize that $A(\overline{A} + B) = AB$ and $A + \overline{A}B = A + B$, you can simplify these expressions more rapidly.

Q45. Simplify the following expressions using the law of common identities.

1. $\overline{B}(E + B)$
2. $K + J\overline{K}$
3. $AB(C + \overline{AB})$
4. $VRS + T\overline{SRV}$
5. $\overline{(XY + Z)}WT + Z + XY$

SUMMARY OF THE BOOLEAN LAWS

1. The law of absorption is:

$A + AB = A$

$A(A + B) = A$.

2. Any expression of a type like A + AB or A(A + B) may be simplified algebraically as follows:

$A(A + B) =$

$AA + AB$

$A + AB =$

$A(1 + B) =$

$A \cdot 1 =$

A

3. The simplest way to use the law of absorption is to convert expressions in A(A + B) form to A + AB form. After completing this step, the next step is to look for a letter or

groups of letters which are ORed together and contain the same letter or group of letters. The more complex group may then be omitted as shown in the example below.

$$BC + C(AB + BD) + DE + CDEF$$

$$BC + \cancel{CAB} + \cancel{CBD} + DE + \cancel{CDEF}$$

$$BC + DE$$

4. The common identities are:

$$A(\bar{A} + B) = AB$$

$$A + \bar{A}B = A + B$$

5. The common identities are not basic laws of Boolean algebra. They are derived from the basic laws, and are useful for simplifying expressions rapidly.

6. Review of the Basic Laws:

The basic laws and common identities of Boolean algebra are:

Identity:	$A = A;\ \bar{\bar{A}} = \bar{A}$
Commutative:	$AB = BA;\ A + B = B + A$
Associative:	$A(BC) = ABC;\ A + (B + C) = A + B + C$
Idempotent:	$AA = A;\ A + A = A$
Double Negative:	$\bar{\bar{A}} = A$
Complementary:	$A\bar{A} = 0;\ A + \bar{A} = 1$
Intersection:	$A \cdot 1 = A;\ A \cdot 0 = 0$
Union:	$A + 1 = 1;\ A + 0 = A$
DeMorgan:	$\overline{AB} = \bar{A} + \bar{B};\ \overline{A + B} = \bar{A}\bar{B}$
Distributive:	$A(B + C) = AB + AC;\ A + BC = (A + B)(A + C)$
Absorption:	$A(A + B) = A;\ A + AB = A$
Common Identities:	$A(\bar{A} + B) = AB;\ A + \bar{A}B = A + B$

Study these laws to be sure you can use them.

You will not always use the same laws in the same order. In general, however, simplification is easiest if you follow this pattern:

1. Remove or split vincula

a. Complementary law-find a group of letters that is the complement of another group.

b. Double negative law and DeMorgan's Theorem-apply one or both, to split or remove vincula.

2. Simplify

a. Apply laws such as these: idempotent, intersection, union, absorption. Simplify as far as possible without changing the form of the expression.

b. Apply the distributive law, or common identities to change the form of the expression. Continue simplifying.

If you can simplify the problems that follow, you have mastered the basic laws of Boolean algebra. If any answer is wrong, or if you feel you could have simplified an expression in fewer steps, compare your work with the steps given.

Q46. $[CD + \overline{F + (\bar{B}\bar{E})} + \overline{(E + B)} + F + C]G$

Q47. $FG + EB\bar{B} + \overline{\bar{G}\bar{F}}$

Q48. $P[KJ + L(N + M) + \overline{J\bar{K}}]$

Q49. $B + C + BD\bar{B}D$

Q50. $(XX + \bar{Y}Y)Z$

Q51. $(\bar{S} + S)RST$

Q52. $(A + \bar{B} + C + \bar{A} + B)(\bar{C} + \bar{A})$

Q53. $RSTU + RSTV + TVWSXR$

Q54. $JK(LMN + \bar{J}\bar{K}PQ)(K + P)$

Q55. $\overline{X\bar{Y}} + \overline{X + Y}$

Q56. $A + B + (C + \bar{B})(E + F)$

Q57. $AC + C + (R + \bar{R})S$

A45. 1. $\overline{B}E$

2. $K + J$

3. ABC

4. $VRS + T$

5. $XY + Z + WT$

A46. $(CD + 1 + C)G =$

$1 \cdot G =$

G

A47. $FG + 0 + GF =$

FG

A48. $PL(N + M) + 1 =$

$P \cdot 1 =$

P

A49. $B + C + 0 =$

$B + C$

A50. $(X + 0)Z =$

XZ

A51. $1 \cdot RST =$

RST

A52. $1(\overline{C} + \overline{A}) =$

$\overline{C} + \overline{A}$

A53. $RST(U + V + VWX) =$

$RST(U + V) =$

$RSTU + RSTV$

A54. $(JKLMN + J\overline{K}\overline{J}PQ)(K + P) =$

$(JKLMN + 0)(K + P) =$

$JKLMNK + JKLMNP =$

$JKLMN + JKLMNP =$

$JKLMN$

A55. $\overline{X} + \overline{Y} + \overline{X}\overline{Y} =$

$\overline{X} + \overline{Y}$

A56. $(A + B + C + \overline{B})(A + B + E + F) =$

$1(A + B + E + F) =$

$A + B + E + F$

A 57. $AC + C + (1 + S) =$

$AC + C + S =$

$C + S$

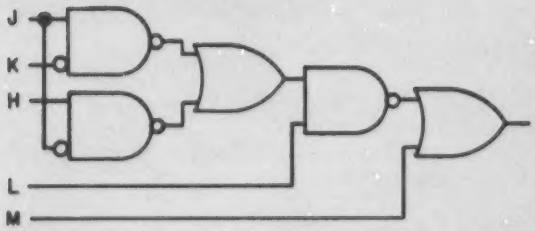
Q58. $\overline{WXY}Z + XW$

Q59. $(B + C)(\overline{D} + B)(\overline{C} + B)$

Q60. For the following diagram, perform three steps.

1. Determine the output expression.
2. Simplify the expression.
3. Diagram the simplified expression.

Then, check your answer.



THE VEITCH DIAGRAM

Now that you have finished the preceding problems using the laws of Boolean simplification, you may have noticed two drawbacks. The first is that some Boolean expressions require many time-consuming steps to simplify. Secondly, unless you are very familiar with all the possible ways the laws of Boolean can be applied, you are never really sure when a Boolean expression is in its simplest form.

Fortunately there is another method you can use for Boolean simplification—the VEITCH DIAGRAM. A Veitch diagram provides a very quick and easy way for finding the simplest logic expression. But, before we go into the construction of Veitch diagrams, it is first necessary to learn some nomenclature, and how to set up a Boolean expression for insertion into a Veitch diagram.

Every character in a Boolean expression has two states that are complements to each other. These complementary states we call variables.

The variable A for example would contain A and $\overline{A}$. If three variables are ANDed together, such as ABC, there are eight possible combinations. These are:

ABC

$AB\overline{C}$

$A\overline{B}\overline{C}$

$\overline{A}B\overline{C}$

$\overline{A}\overline{B}C$

$\overline{A}BC$

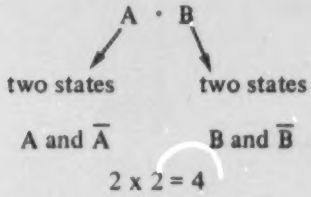
$\overline{A}B\overline{C}$

$A\overline{B}C$

These possible combinations are called minterm. To be classified as a minterm, an expression MUST be the AND product of an expression. To illustrate, the variables A, B, and C to be minterm must be expressed as ABC.

Because each variable has two possible states, the number of combinations can be expressed as the number of possible states (two) raised to a power dictated by the number of variables.

For example an expression with two variables (A and B) will have four minterms or 2^2.



If an expression contains three variables:

ABC

$2 \times 2 \times 2 = 8$

and, finally, if it contains four variables:

$ABCD$

$2 \times 2 \times 2 \times 2 = 16$

Q61. Which of the following expressions are minterm?

1. $A + B + C$
2. $CBDAE$
3. $A + \overline{A} + C$
4. $ADCB$

Before expressions can be plotted on a Veitch diagram they must be placed in minterm form. What this means is the expression MAY NOT contain parentheses or a vinculum extending for more than one letter. In addition, each minterm must be separated by an OR sign. The following are minterm form expressions.

$AB + CD + ABE$

$A + ABC + DE$

The following examples are non-minterm form expressions.

1. $\overline{T + V} + RS$
2. $A + B + \overline{CD} + \overline{A}BC$
3. $\overline{W}\overline{X}YZ + X(W + \overline{Y})$
4. $A(B + C + \overline{D}) + E + AB\overline{C} + \overline{\overline{D}}C\overline{E}$
5. $ABC + \overline{W\overline{X}}XW + MNOM$

If we desired to use the Veitch diagram for the Boolean simplification of the preceding non-minterm form expressions, they will first have to be converted to minterm form. Fortunately, any expression can be converted to minterm form by the following process.

- Split or remove vinculum/vincula by using DeMorgan's theorem.

- Remove parentheses by using the distributive law.

A58. $(\overline{WX} + \overline{Y})\,Z + WX$

$\overline{WX}Z + WX + \overline{Y}Z$

$WX + Z + \overline{Y}Z$

$WX + Z(1 + \overline{Y})$

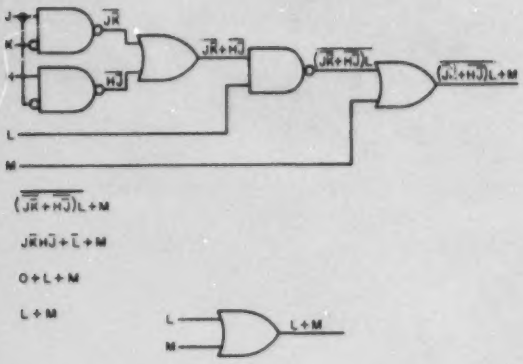
$WX + Z$

A59. $B + C\overline{D}\overline{C} =$

$B + 0 =$

B

A60.



A61. 2 and 4

- Simplify within the term. For example:

ABCA→ABC

or

$ABC\overline{A} + DE \rightarrow 0 + DE \rightarrow DE$

Once an expression is in minterm form do not attempt to simplify further.

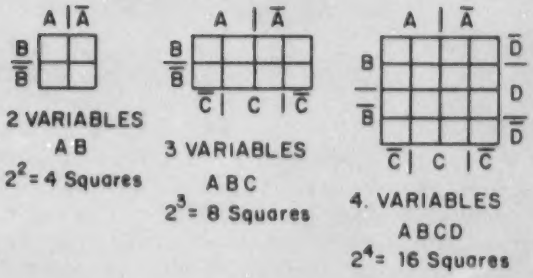
Q62. **Convert the five examples of non-minterm form expressions, which are given in the preceding listing, to minterm form.**

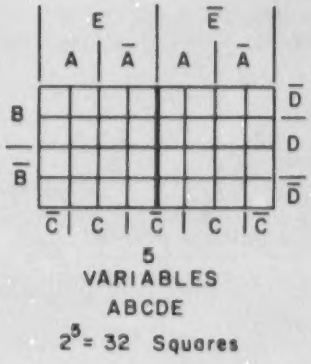
Once an expression is in minterm form, it is ready to be set up on the Veitch diagram. The Veitch diagram is constructed in the following manner.

Step 1 Count the number of different variables in the expression.

Step 2 The total number of different variables derived in step 1 is used as a power of two (e.g. 2^3).

Step 3 Draw a Veitch diagram containing the number of boxes equal to the power of two derived in step 2.



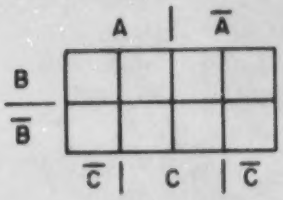


Step 4 Label the Veitch diagram such that each variable covers half the total number of squares either horizontally or vertically and overlaps every other variable with the exception of its complement, as shown above.

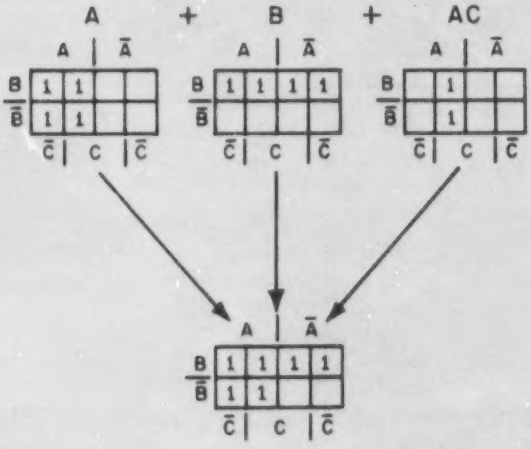
At this time it should be pointed out that, while Veitch diagrams for more than four variables are still valid, you would probably spend more time drawing squares than you would if you simplified the expression using conventional Boolean means. However, since most Boolean expressions usually contain four or fewer variables, this hardly matters.

Now that you have learned how to place expressions in minterm form and construct a Veitch diagram, we will discuss the manner in which the expressions are plotted on the Veitch diagram.

To plot the expression $A + B + AC$, first determine the size of the Veitch diagram needed then draw and label it as shown below.



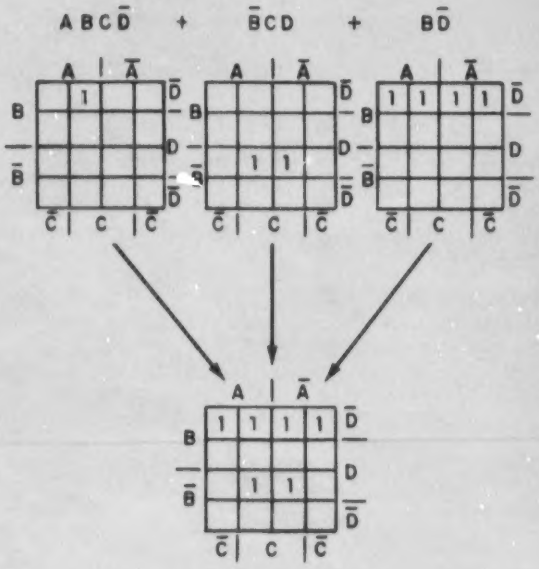
Next, start with the first minterm on the left and place a 1 in each block on the Veitch diagram in which all the variables for that minterm overlap. If a minterm contains only one variable, mark all the squares that contain that one variable. Continue on until the entire expression is plotted. If a square is already 1'ed, do not double mark it. This is shown below.



Q63. Plot the following expressions.

1. $\bar{A}\bar{B}\bar{C} + A\bar{B}C + A$
2. $BC + A$
3. $A + \bar{A}B + C$

The 16- and 32-square Veitch diagrams are plotted in the same manner as the 8-square. The following illustrations are examples of 16-square Veitch diagrams for four variables.



Q64. Plot the following expressions.

1. $A(\bar{B}C + CA) + AB$
2. $WXY + Z + \overline{W + Y}$
3. $MNO + O\bar{P} + MN$

You have learned to plot minterm form expressions on the various Veitch diagrams. Now, you are going to learn to extract the simplest expression from the plot. We will start

A62. 1. $\overline{T} \cdot \overline{V} + RS$

2. $A + B + \overline{C} + \overline{D} + \overline{A}C + \overline{B}C$

3. $\overline{WXYZ} + WX + X\overline{Y}$

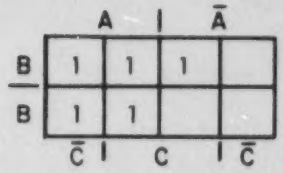
4. $AB + AC + A\overline{D} + E + AB\overline{C} + \overline{D}C\overline{E}$

5. $ABC + \overline{WXXW} + MNOM \rightarrow ABC + MNO$

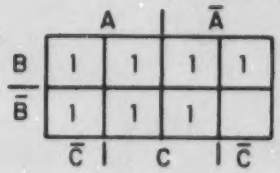
A63.

1.

2.

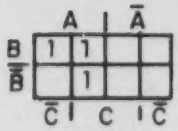


3.

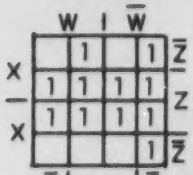


A64.

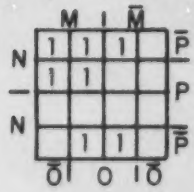
1.



2.



3.



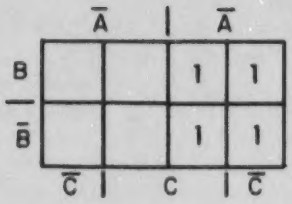


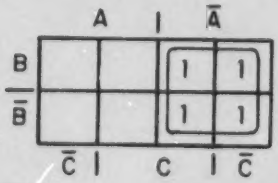
Figure 2-8.—Order of Preference for a Three-variable Expression.

with the three-variable expression $\bar{A}BC + \bar{A}B\bar{C} + \bar{A}\bar{B}$, plotted as shown in figure 2-8. Next, we extract the simplified expression by looking for patterns in the order of preference shown in figure 2-8 and described below.

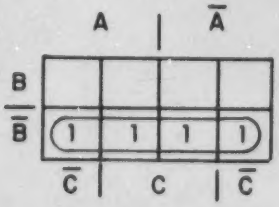
- Four squares joined together will describe a one-variable term.
- Two squares joined together will describe a two-variable term.
- One square that stands alone will describe a three-variable term.

Join all the adjacent squares into as large a grouping as possible—four squares, two squares, or one square. Write the expression that is common to all squares of each group.

In our example, only $\bar{A}$ is common to all the squares grouped together. Therefore, $\bar{A}$ is the simplified expression for $\bar{A}BC + \bar{A}B\bar{C} + \bar{A}\bar{B}$.



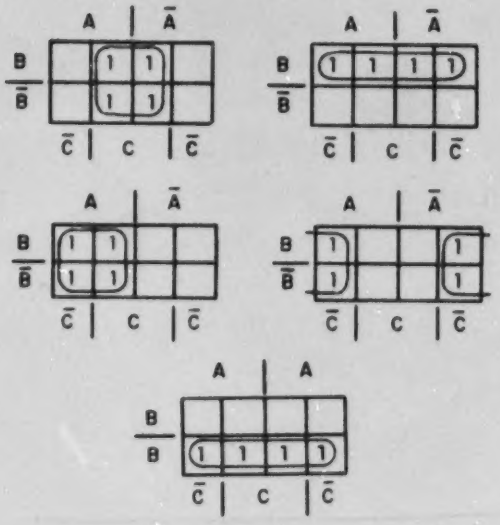
$\bar{B}C + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C}$ is plotted as



The four squares are adjacent to each other and can therefore be grouped together. From the order of preference given in figure 2-8, we find that one term will replace the expression contained in the four squares.

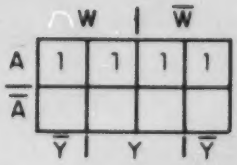
The only term common to all four squares is $\bar{B}$, therefore $\bar{B}$ is the simplified form of $\bar{B}C + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C}$.

Squares may be grouped in groups of four in the following way.

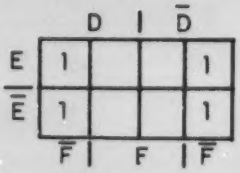


Q65. Describe the plots below.

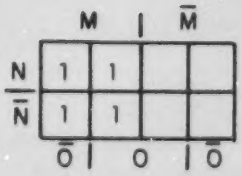
1.



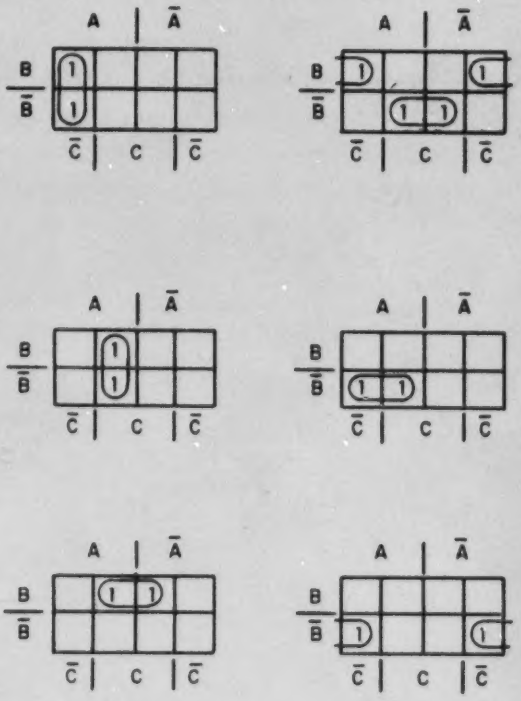
2.



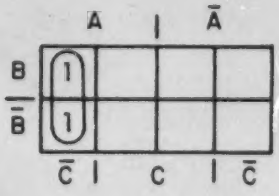
3.



When an expression can not be simplified to a single-variable expression, you must try to simplify the expression to two-variable terms. This is done by linking the squares into groups of two. Some of the possible patterns for groups of two are:



If $AB\bar{C} + A\bar{B}\bar{C}$ is plotted



and then the simplified expression extracted, we get $A\bar{C}$. $A\bar{C}$ then is the simplified form for $AB\bar{C} + A\bar{B}\bar{C}$.

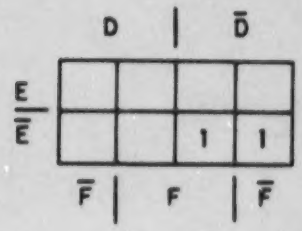
A65. 1. A

2. $\overline{F}$

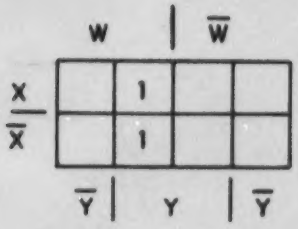
3. M

Q66. **Describe the following plots as simply as possible.**

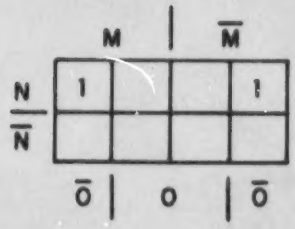
1.



2.



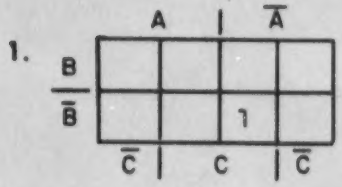
3.



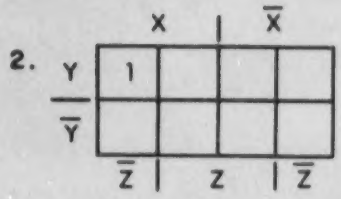
When the squares can not be grouped in groups of four or two, then the terms must form single squares. Remember, a single square on a three-variable Veitch diagram describes a three-variable term. Just write all the terms common to the single square.

Q67. **Describe the following plots as simply as possible.**

1.

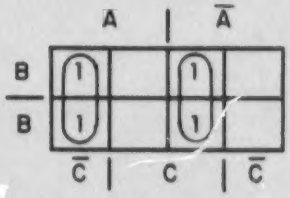


2.



In each of the previous diagrams, all the plotted squares could be identified with one minterm. Often it is necessary to use more than one minterm to describe a plot. In these cases, the terms must be joined with OR signs.

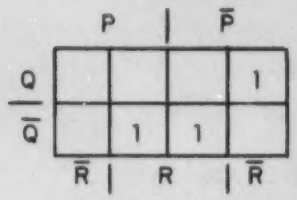
$A\overline{B}\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + \overline{A}\overline{B}C$ is plotted as



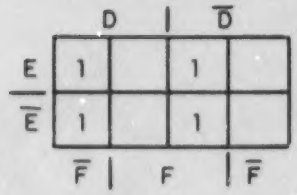
These squares are described by the expression $A\overline{C} + \overline{A}C$.

Q68. Describe the following plots:

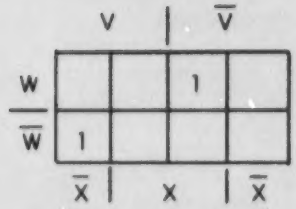
1.



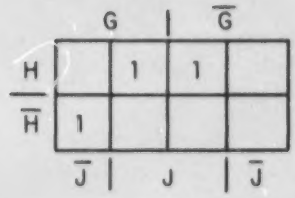
2.



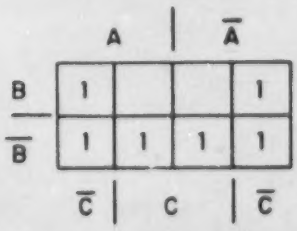
3.



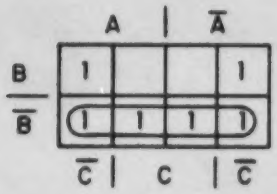
4.

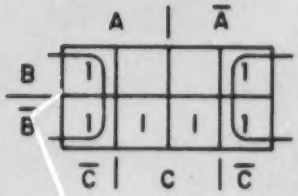


So far, all the minterms used have been described by groups of squares that are separated from each other; but, how do you describe something like the following?



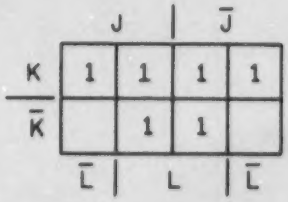
First, try to group all the squares into groups of four, if possible.



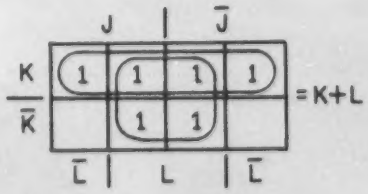


You will notice that some squares were used more than once. This is permissible. The simplified expression is $\overline{B} + \overline{C}$.

Let's try another example.



What is the simplified expression?



A66. 1. $\bar{D}\bar{E}$

2. WY

3. $N\bar{O}$

A67. 1. $\bar{A}\bar{B}C$

2. $XY\bar{Z}$

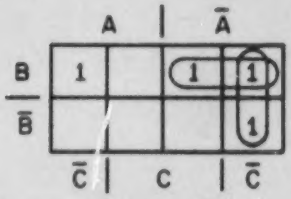
A68. 1. $\bar{Q}R + \bar{P}Q\bar{R}$

2. $D\bar{F} + \bar{D}F$

3. $\bar{V}WX + V\bar{W}\bar{X}$

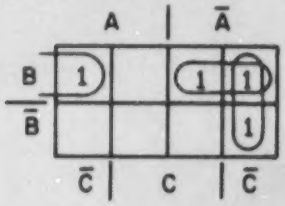
4. $HJ + G\bar{H}J$

This brings up an important point, whenever you simplify using a Veitch diagram, always look closely for possible combinations of blocks. What is obvious is not always right. For example, the following diagram appears to describe three minterms, two of two variables each and one of three variables.



$AB\bar{C} + \bar{A}B + \bar{A}\bar{C}$

But, in the correct method, the single square to the left can be grouped to the one on the right to describe an expression of three minterms consisting of two variables each.



$\bar{A}\bar{C} + \bar{A}B + B\bar{C}$

Q69. **Simplify the following expressions.**

1. $JK\bar{L} + JKL + J\bar{K}\bar{L} + \bar{J}KL + \bar{J}K\bar{L} + \bar{J}\bar{K}\bar{L}$

2. $XY\bar{Z} + X\bar{Y}\bar{Z} + \bar{X}\bar{Y}Z + \bar{X}YZ + \bar{X}Y\bar{Z} + \bar{X}\bar{Y}\bar{Z}$

3. $MN\bar{P} + M\bar{N}\bar{P} + M\bar{N}P + \bar{M}\bar{N}\bar{P}$

4. $\bar{B}CD + B\bar{C}\bar{D} + \bar{B}C + BC\bar{D} + B\bar{C}D + BC$

SUMMARY OF THREE-VARIABLE VEITCH DIAGRAMS

1. An expression can often be simplified most efficiently by plotting it on a Veitch diagram, then extracting the simplified expression from the plot.

2. To plot an expression on a Veitch diagram, you must convert it to minterm form.

3. To convert an expression to minterm form,

   a. split or remove vincula ($\overline{\bar{A} + \bar{B}} = \bar{A}\bar{B}$; $\bar{\bar{A}} = \bar{A}$),

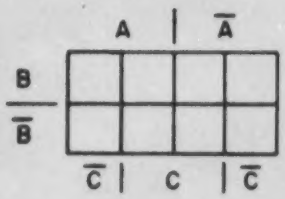
   b. remove parentheses (distributive law), and

   c. simplify within the term ($ABCA = ABC$; $ABC\bar{A} = 0$).

4. Determine the number of squares needed in the Veitch diagram by using the number of variables as a power of two.

   3 variables—2^3 = 8 squares
   4 variables—2^4 = 16 squares
   5 variables—2^5 = 32 squares

5. A Veitch diagram for variables A, B, and C is labeled:



6. Half of the squares are assigned to each variable, and the other half to its complement. Each variable overlaps every other variable, and every complement but its own.

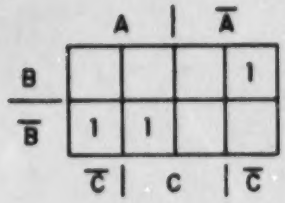
7. Plot one term at a time on the diagram. On an eight-square Veitch diagram,

a. a one-variable term (A) occupies four squares,

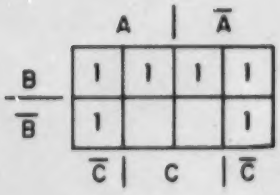
b. a two-variable term ($\bar{A}\bar{B}$) occupies two squares, and

c. a three-variable term ($A\bar{B}C$) occupies one square.

$A\bar{B} + \bar{A}B\bar{C}$ would be plotted:



8. The plots for two or more terms may overlap. $B + \bar{C}$ would be plotted:



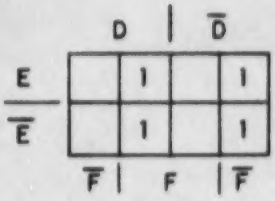
9. To simplify an expression, describe the plot with as few terms as possible. Look for patterns of plotted squares in this order:

1. Four plotted squares described by a one-variable term

2. Two plotted squares described by a two-variable term

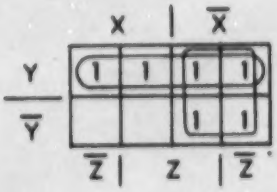
3. One plotted square described by a three-variable term

$DEF + \bar{D}\bar{F} + D\bar{E}F$ is plotted:



The simplified expression is $DF + \bar{D}\bar{F}$.

10. Also, describe the plot with as few variables as possible. $\bar{X}$ and Y overlap in the plot below. By using two squares twice, you can describe the plot as $\bar{X} + Y$. This is preferable to $\bar{X} + XY$, or $Y + \bar{X}Y$.



The simplified expression is extracted from the 16-square Veitch diagram in the same manner as it was in the 8-square Veitch diagram. Now you should look for the following patterns:

1. Eight plotted squares describe a one-variable term

2. Four plotted squares describe a two-variable term

A69. 1. $K + \overline{L}$

2. $\overline{X} + \overline{Z}$

3. $M\overline{P} + M\overline{N} + \overline{N}\overline{P}$

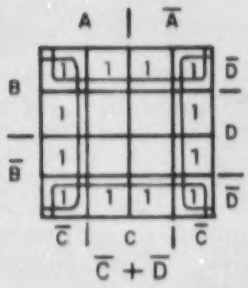
4. $B + C$

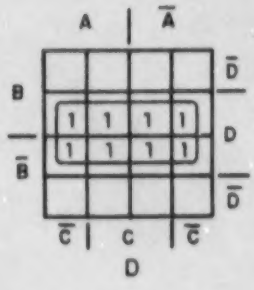
---

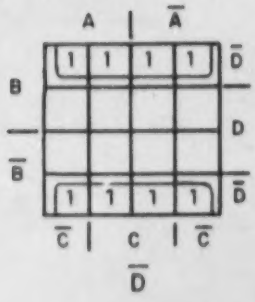
3. Two plotted squares describe a three-variable term

4. One plotted square describes a four-variable term

You can spot the patterns quickly if you recognize that patterns are formed by adjacent squares and by squares at opposite ends of rows. For example:

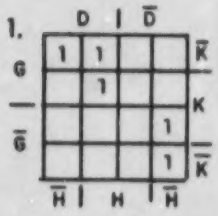




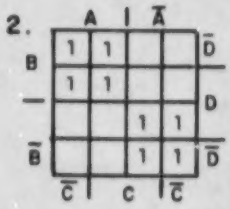


Q70. Extract the simplified expression from each of the following Veitch diagrams.

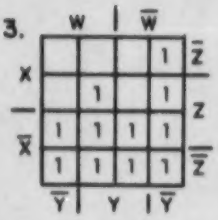
1.



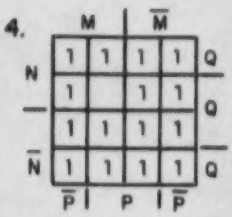
2.



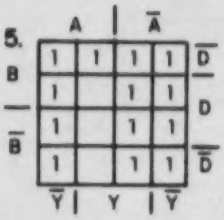
3.



4.

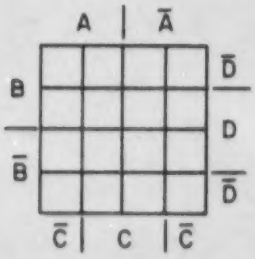


5.



SUMMARY OF THE FOUR-VARIABLE VEITCH DIAGRAM

1. A Veitch diagram for four variables, A, B, C, and D, is labeled:



2. As stated before, half the squares are assigned to each variable, and half to its complement. Each variable overlaps every other variable, and every complement but its own.

3. On a 16-square Veitch diagram,

   a. a one-variable term (A) occupies eight squares,

   b. a two-variable term ($\bar{A}\bar{B}$) occupies four squares,

   c. a three-variable term ($\bar{A}B\bar{C}$) occupies two squares, and

   d. a four-variable term ($A\bar{B}C\bar{D}$) occupies one square.

4. To describe the plot of a 16-square Veitch diagram, look for:

   1. Eight plotted squares described by a one-variable term

   2. Four plotted squares described by a two-variable term

   3. Two plotted squares described by a three-variable term

   4. One plotted square described by a four-variable term

5. Patterns of plotted squares are formed by adjacent squares or by squares at opposite ends of rows.

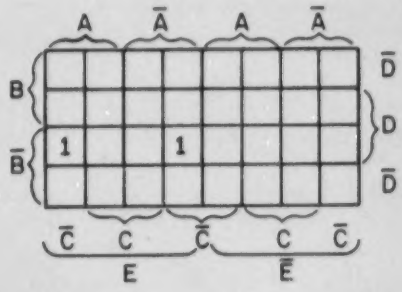
THE THIRTY-TWO-SQUARE VEITCH DIAGRAM

The simplified expression of a 32-square Veitch diagram can be extracted by looking for the following patterns:

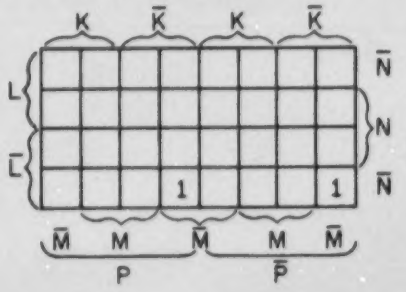
1. Sixteen squares describe a one-variable term
2. Eight squares describe a two-variable term
3. Four squares describe a three-variable term
4. Two squares describe a four-variable term
5. One square describes a five-variable term

In the 8- and 16-square Veitch diagram, you looked for patterns in adjacent squares and at opposite ends of rows. The 32-square Veitch has an additional pattern; namely, at corresponding positions in the same horizontal row.

To understand this, think of the 32-square diagram as two 16-square diagrams. Now, the plotted squares below are at opposite ends of a row. Describe them together as $\bar{B}CDE$.



Plotted squares may be described together if they are in the same position on both 16-square blocks. If the right half were superimposed on the left half, the plotted squares would overlap. Describe them together as $\bar{K}L\bar{M}\bar{N}$.



A70. 1. $DG\bar{K} + DGH + \bar{D}\bar{G}\bar{H}$

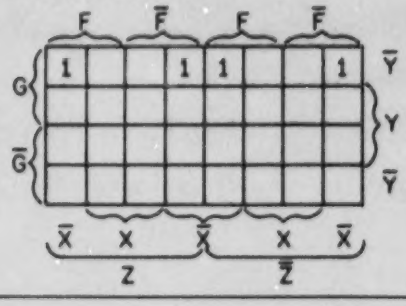
2. $AB + \bar{A}\bar{B}$

3. $\bar{X} + \bar{W}\bar{Y} + WYZ$

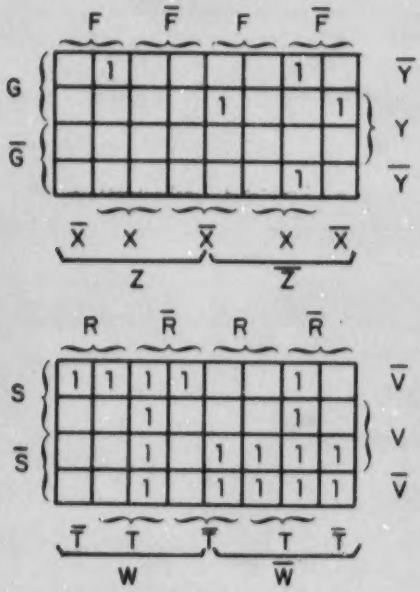
4. $\bar{M} + \bar{N} + \bar{P} + \bar{Q}$

5. $\bar{A} + B\bar{D} + \bar{Y}$

The following plot is a combination of the two situations above. By superimposing one half on the other, you have squares at opposite ends of a row. The four squares can be described by a three-variable term, $G\bar{X}\bar{Y}$.



Q71. Describe the following plots.



SUMMARY OF BOOLEAN ALGEBRA

A summarization of the main points contained in this topic follows.

The information in this topic is based upon the assumption that most quantities have only two possible states, either TRUE or FALSE. This assumption is called Boolean Algebra or Boolean Logic. Boolean is a description of the input conditions necessary to get a desired output from a logic circuit.

A TRUTH TABLE–Used to summarize the logic circuit and can be used to describe the input conditions necessary to obtain a desired output from a logic gate.

INPUTS		OUTPUTS
A	B	f
0	0	0
0	1	0
1	0	0
1	1	1

TRUTH TABLE

THE BOOLEAN EXPRESSION–A written description of the logic in a circuit. e.g. A$\overline{B}$

THE VINCULUM–The straight horizontal line or lines which are placed above a letter or letters in a Boolean expression to indicate negation and can serve as a sign of grouping.

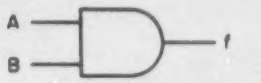
$\overline{\overline{A}} = A$

$\overline{A\overline{B}} = \overline{A} + \overline{\overline{B}}$

$\overline{(A+B)} = \overline{A}\,\overline{B}$

INPUTS		OUTPUTS
A	B	f(AB)=AB
0	0	0
0	1	0
1	0	0
1	1	1

THE LOGIC AND GATE–In topic 1, it was pointed out that the AND gate is configured so that all inputs must be a TRUE or binary 1 to get a TRUE or binary 1 output. In this topic it was shown that when a statement is negated it will be equal to its complement. This was shown by the use of a vinculum placed above the variable letter. The AND function was indicated by placing a dot between letters or simply by grouping the terms together.



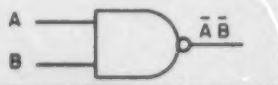
$\overline{(A+B)} = \overline{A} \cdot \overline{B}$ or $\overline{A}\,\overline{B}$

THE LOGIC OR GATE–The logic OR gate function is indicated by placing a plus sign between terms. You learned that only one input need be TRUE (1) to get a TRUE (1) output.

INPUTS		OUTPUTS
A	B	f(A,B)A+B
1	1	1
1	0	1
0	1	1
0	0	0

A71. 1. $\bar{F}X\bar{Y}\bar{Z} + G\bar{X}Y\bar{Z} + FGX\bar{Y}Z$

2. $\bar{S}\bar{W} + \bar{R}T + S\bar{V}W$



A	B	AB	$f(A,B) = \bar{A}\bar{B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

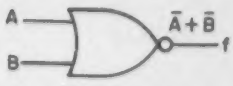
THE LOGIC NAND GATE–The logic NAND gate function can be expressed as an AND gate with an inverter on its output. The vinculum, or bar, is used as a grouping symbol to indicate this inversion.

A	B	A+B	$f(A,B) = \overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

THE LOGIC NOR GATE–The logic NOR gate functions in the same manner as a NAND gate, except that its parent gate is a basic OR gate with an inverter on its output.

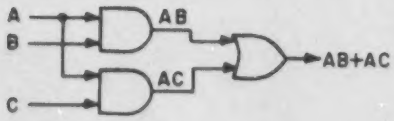
A	$f(A) = \bar{A}$
1	0
0	1

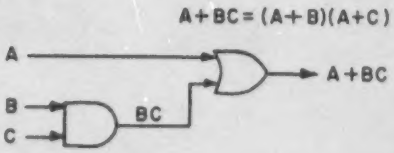
THE LOGIC NOT FUNCTION–The NOT function is an inverter which is placed either on the input or the output side of a logic gate. An example of this would be to place the NOT function on the output of the AND gate to form the NAND gate or to place the NOT function on the output of an OR gate to form the NOR gate. Another purpose of the NOT function is to invert the input signal to any logic gate.



THE OUTPUT EXPRESSION OF LOGIC DIAGRAMS–The following facts should be remembered when diagraming the output expression of a logic diagram.

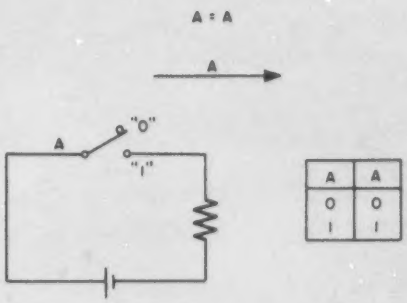
- A logic diagram is composed of two or more logic symbols.
- If a logic symbol is at the extreme left of a diagram, its inputs are single letters.
- Parentheses are used to indicate grouping.
- The vinculum is used to group the output expressions that have been inverted.



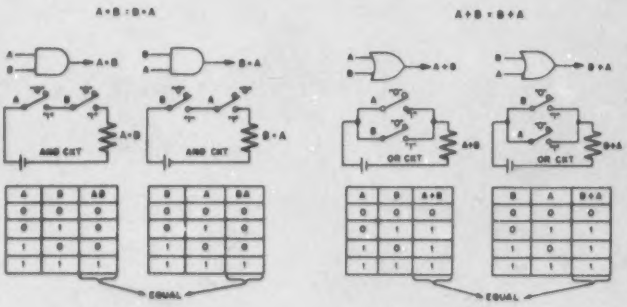


BOOLEAN LAWS AND THEOREMS

THE LAW OF IDENTITY stated that any expression is equal to its self e.g. A = A.



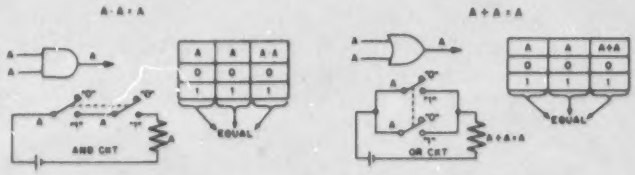
THE COMMUTATIVE LAW states that when logic symbols are ANDed or ORed, together, the order in which they are written does not affect their value e.g. ABC = CAB.



THE ASSOCIATIVE LAW states that expressions such as A(BC) or A + (B + C) can be simplified by rewriting and rediagraming.

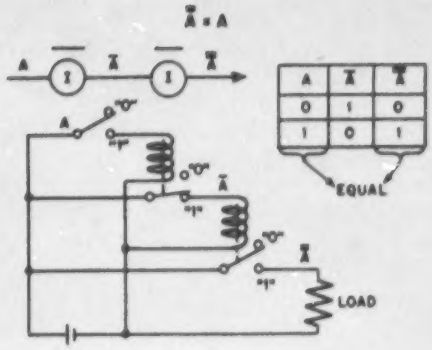
A (BC) = ABC
A + (B + C) = A + B + C

THE IDEMPOTENT LAW states that combining a quantity with itself either by logical addition or logical multiplication will result in a logical sum or product that is the equivalent of the quantity. This can be stated as A + A + A = A or AAA = A.

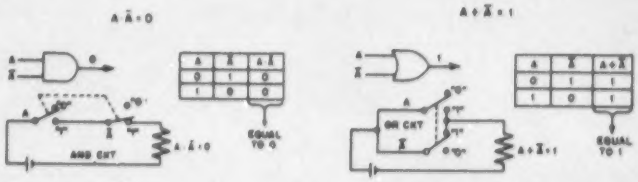


THE DOUBLE NEGATIVE LAW is used to aid simplification in logic expressions which have a number of vincula contained in them. Remember that if there is an even number of vincula, write that portion of the expression as a non-NOT function; and if there is an odd number of vincula, that portion of the expression will be written as a NOT function.

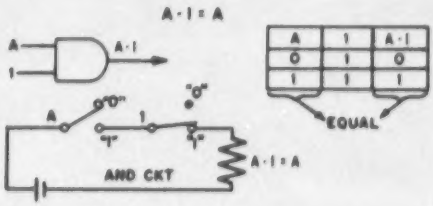
e.g. $\overline{\overline{A}B} + \overline{\overline{C}} = \overline{A}B + C$



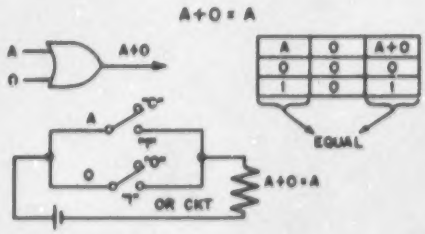
THE COMPLEMENTARY LAW may be stated for the logic AND and OR gate functions in an expressionary form as $A\overline{A}$ = FALSE (0) or $A + \overline{A}$ = TRUE (1). In other words an AND function requires all inputs be TRUE to get a TRUE output and an OR function requires that only one input be TRUE to get a TRUE output.



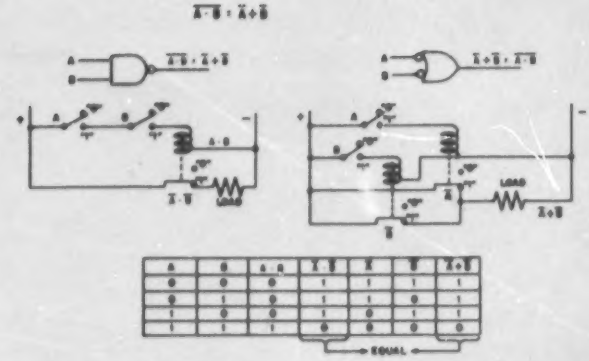
THE LAW OF INTERSECTION states that if one input to an AND gate is already TRUE, then the output will depend upon the state of the other inputs only.

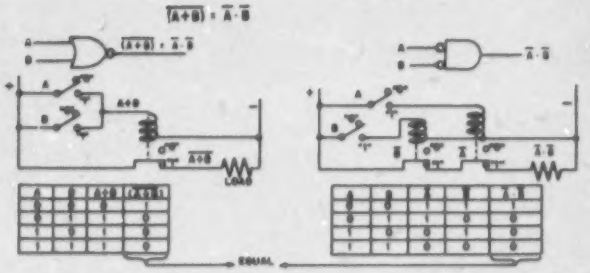


THE LAW OF UNION is the same as the law of intersection except that it applies to the OR gate function. That is, if one input is already FALSE the only way to get a TRUE output is if one of the other inputs is TRUE.

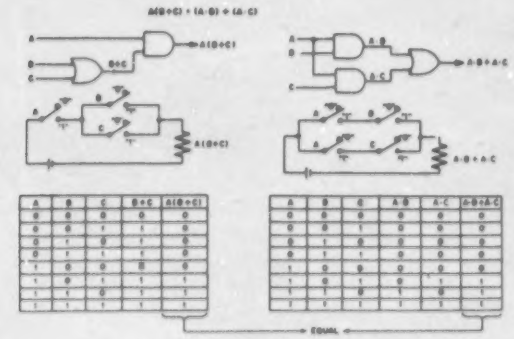


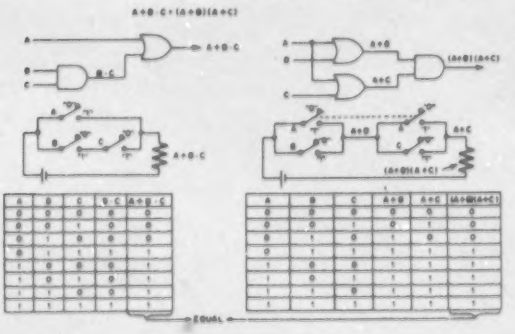
DeMorgan's THEOREM is concerned with NAND, NOR, and NOT functions, with it you can split or join vincula. The first part of the theorem deals with NAND functions and states that $\overline{AB} = \bar{A} + \bar{B}$. The second part of the theorem deals with NOR functions and states that $\overline{A + B} = \bar{A}\bar{B}$.

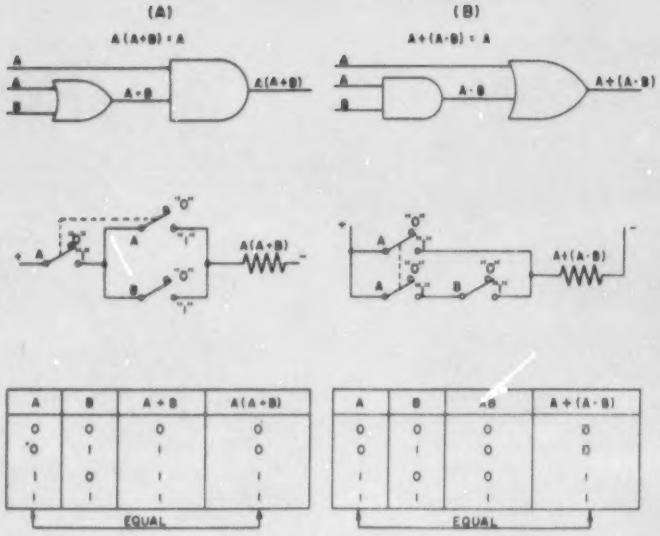




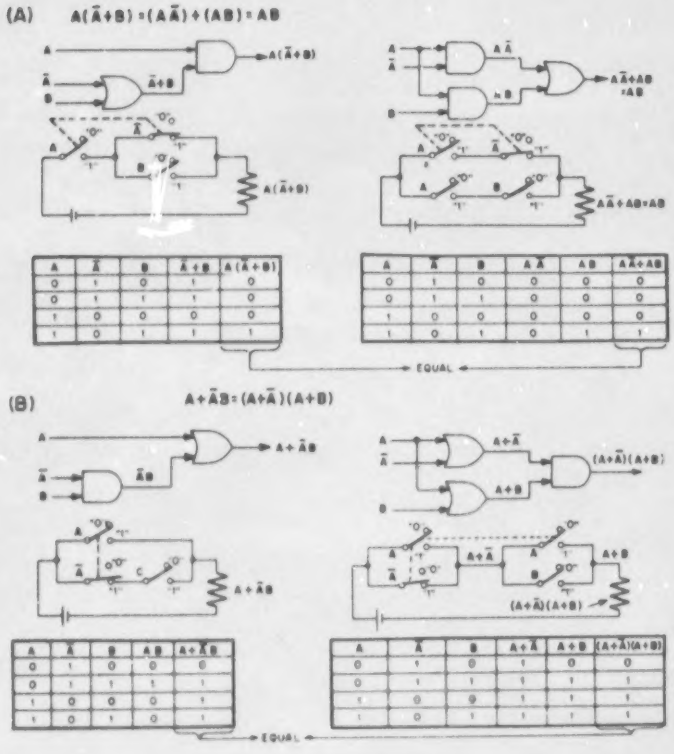
THE DISTRIBUTIVE LAW is an application of normal algebra, in that it states that $A(B + C) = AB + AC$. This law is used to manipulate a logic expression so that one of the other laws can be used to simplify.







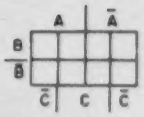
THE LAW OF ABSORPTION is another of the laws which is used to manipulate a logic expression to aid in simplification. The law of absorption states that A(A + B) = A or A + AB = A. This in effect says that anytime that you have an A you will get an A output.



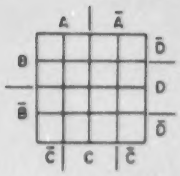
THE LAW OF COMMON IDENTITIES is a law which governs the most frequently occuring Boolean expressions, that would normally be simplified by applying a combination of other Boolean laws. Once these identities are learned, they increase the speed of simplification. The law of identities states that anytime the expression A(Ā + B) = AB or A + ĀB = A + B appears it can immediately be simplified to AB without going through the process of using the distributive, complementary, or law of union to simplify.

THE VEITCH DIAGRAM is used in finding the simplest logic equation needed to express a given function. This simplification method is based on the fact that a Boolean expression has two states that are complimentary to each other. Any number of variables may be plotted on a Veitch diagram, though the diagrams become difficult to use when more than four variables are involved.

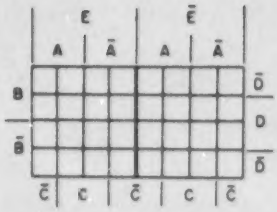
2 VARIABLES



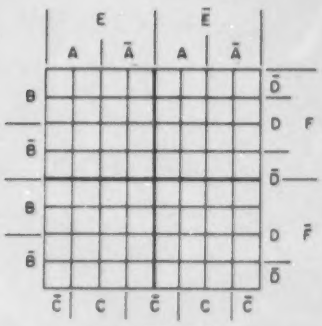
3 VARIABLES



4 VARIABLES



5 VARIABLES



6 VARIABLES

Table 2-1.—Boolean Laws and Theorems

1.	Law of Identity	$A = A$
		$\overline{A} = \overline{A}$
2.	Commutative Law	$AB = BA$
		$A + B = B + A$
3.	Associative Law	$A(BC) = ABC$
		$A + (B + C) = A + B + C$
4.	Idempotent Law	$AA = A$
		$A + A = A$
5.	Double Negative Law	$\overline{\overline{A}} = A$
6.	Complementary Law	$A\overline{A} = 0$
		$A + \overline{A} = 1$
7.	Law of Intersection	$A \cdot 1 = A$
		$A \cdot 0 = 0$
8.	Law of Union	$A + 1 = 1$
		$A + 0 = A$
9.	DeMorgan's Theorem	$\overline{AB} = \overline{A} + \overline{B}$
		$\overline{A + B} = \overline{A}\,\overline{B}$
10.	Distributive Law	$A(B + C) = AB + AC$
		$A + BC = (A + B)(A + C)$
11.	Law of Absorption	$A(A + B) = A$
		$A + AB = A$
12.	Law of Common Identities	$A(\overline{A} + B) = AB$
		$A + \overline{A}B = A + B$

BEST COPY AVAILABLE

TOPIC 3

LOGIC CIRCUITS

Most computer functions are based on the operation of just seven basic circuits: the bistable multivibrator (or flip-flop), the AND, OR, NAND, NOR, and Exclusive OR gates, and the NOT function (or INVERTER). Only the FIVE gates and the NOT function will be covered in this topic.

LOGIC SWITCHING

Topic 3 covers the electronic theory necessary to understanding logic gates. A logic gate circuit is an electronic switch that has two operating states—either all ON or all OFF. Each type of electronic switch (logic gate circuit) has a set of conditions peculiar to itself that will either turn it on or off. When the conditions have been met to turn on a switching circuit, the circuit is said to be "gated on," and when the switch is turned off, it is said to be "gated off." The set of conditions that controls the gating of a logic gate circuit is called its LOGIC. A logic gate is the collective term for each type of electronic switch along with the logic required to gate it either on or off. These principles are illustrated in figure 3-1.

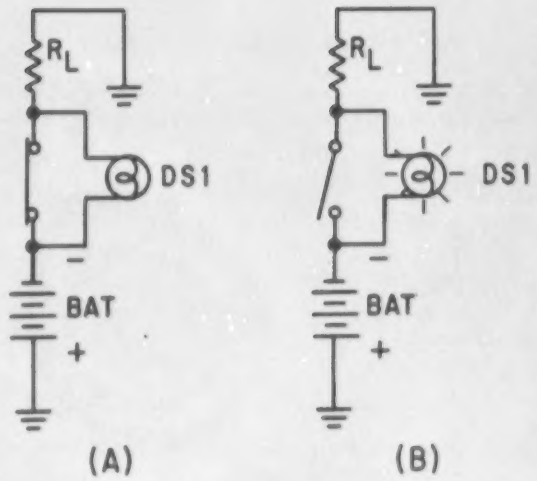


Figure 3-1.—Logic Switches.

In figure 3-1(A), current flows through the switch to ground through the load; therefore, the lamp is not illuminated. In figure 3-1(B), the switch is open and current flows through the lamp, causing it to illuminate. The lamp was "gated on" by the switch; the condition that caused the switch to open is its logic (in this case the author's pen). Because electronic computers, and not manual computers, are being discussed, the operating principles of an electronic switch will be presented. An example of a logic gate that you are already familiar with is the class C biased amplifier. In figure 3-2, the manual switch (used in figure 3-1) has been replaced with a class C biased transistor (Q1). You will recall that the class C amplifier is biased below cutoff, and will stay in this condition until a signal of sufficient amplitude and polarity causes it to be driven into conduction. Note in

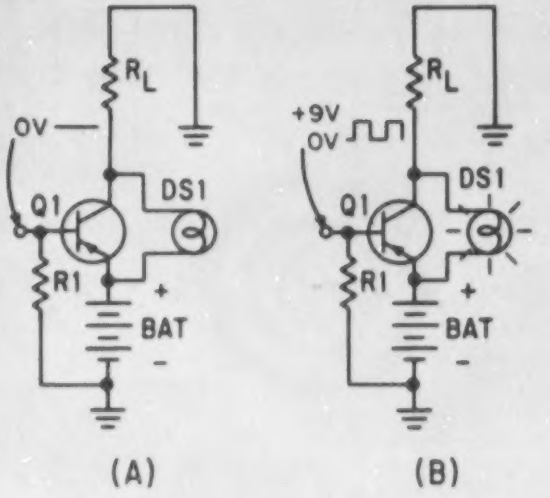


Figure 3-2.—Electronic Logic Switch with Positive Pulse.

figure 3-2(A) that no signal is being applied to the base, and in figure 3-2(B) a series of positive pulses is being applied to the base. These pulses are of sufficient amplitude to drive the amplifier from cutoff directly to saturation and from saturation to cutoff.

In figure 3-2(A) the transistor is biased into saturation by the negative battery voltage on its base. There is maximum current flowing through the transistor (or switch) and the lamp is not illuminated. In figure 3-2(B), each time a +9 volt pulse is applied to the transistor it causes Q1 to go into cutoff. During cutoff, current flows from the battery to ground to the load resistor (R_L) and through the lamp to the positive terminal of the battery. Thus, the lamp in figure 3-2(B) is "gated on" by the +9 volt pulse. The +9 volt pulse is therefore the logic for the gating circuit. To put it another way, the +9 volt pulse is the condition required (logic) to "gate on" the lamp. The principles of gating and logic will be used throughout the rest of this topic. It should be pointed out that transistors used in computer applications are of a special type, called a SWITCHING TRANSISTOR. The only difference between a switching transistor and the type you are familiar with is that the base of a switching transistor is extremely thin. This eliminates all class A operation. The transistor can be operating in only one of two states, either cutoff or saturation. This makes a switching transistor the perfect switch because it has only two states: open or closed. When the transistor is in saturation, it is effectively a short circuit.

In figure 3-3(A) the transistor is in its quiescent (no input signal) state, with the base positive with respect to the emitter. This causes the transistor to be saturated, shorting out the lamp. In figure 3-3(B), each time a -9 volt pulse is applied to the base, it causes the transistor to cut off. Current then flows from the source through the lamp to the the load, causing the lamp to illuminate. Thus, the lamp is "gated on" by the -9 volt pulse. After the gating pulse is no longer present, the transistor returns to saturation, switching off the lamp. In this example, the -9 volt pulse is the logic for the gate.

LOGIC POLARITY

You should have noticed in the previous examples that one of the electronic switches was caused to change state by a positive pulse (fig. 3-2), and the other by a negative pulse (fig. 3-3).

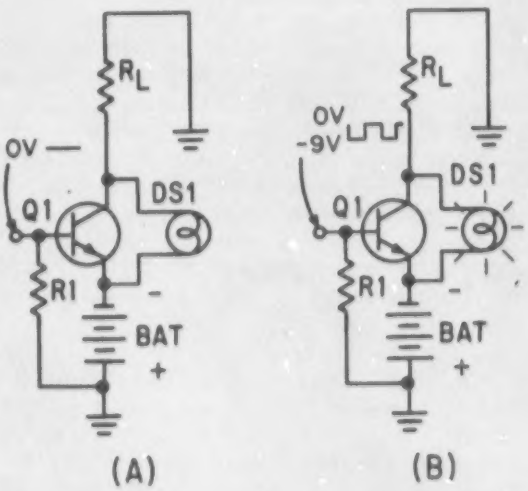


Figure 3-3.—Electronic Logic Switch with Negative Pulse.

These two examples demonstrate the POLARITY OF LOGIC.

If the logic pulse that causes an electronic switching element of a logic circuit to change from the quiescent (no signal input) state to the opposite state is positive-going, then the logic is POSITIVE LOGIC. If a negative-going pulse causes an electronic switch to change from the quiescent state to the opposite state, it is NEGATIVE LOGIC. This is illustrated in figure 3-4. You should take note that the quiescent voltage is NOT always zero. For example, the quiescent state in each of the four examples below is at the 9-volt level.

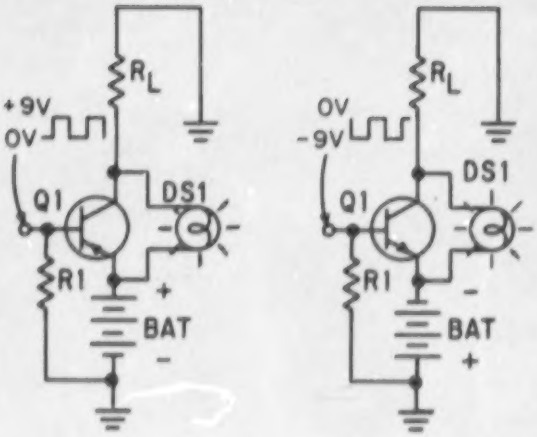
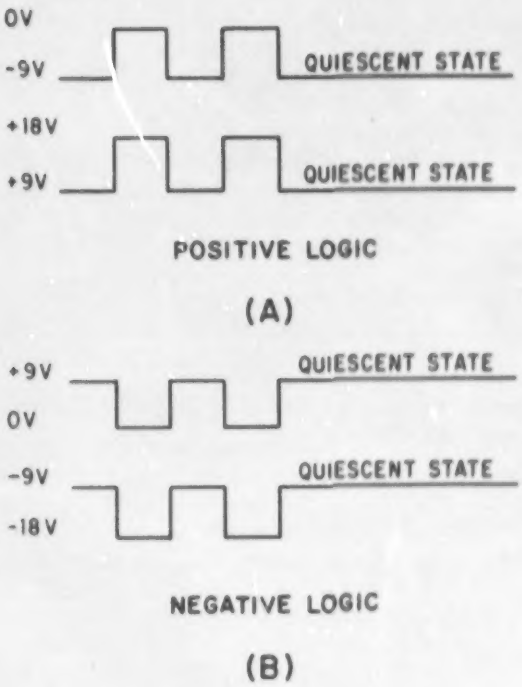


Figure 3-4.—Positive and Negative Logic.

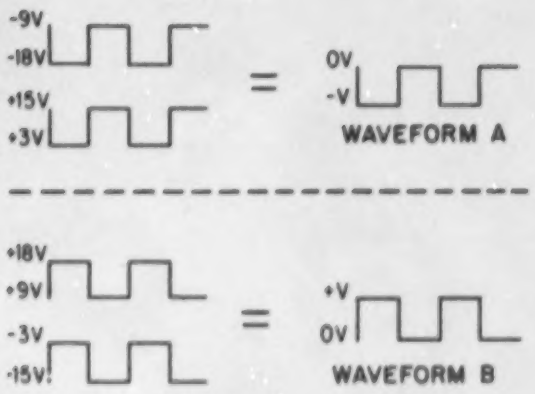


In (A) the pulses going from -9 volts to 0 volts and from +9 volts to +18 volts are positive-going pulses and can cause an electronic switch configured for positive logic to go from the quiescent state to the opposite state. The same principle applies in (B) where the pulses going from -9 volts to -18 volts and from +9 volts to 0 volts are considered to be negative logic.

If given a logic circuit and told the polarity of logic of the circuit, you can diagram the train of logic pulses inputted to the circuit. Keep in mind that, theoretically, 0 volts has an "electrical potential" when used in a logic circuit. The polarity of 0 volts is relative to the polarity of the pulses used with it.

In summary, the polarity of logic of any circuit is determined by the relative polarities of the quiescent state and the level of the pulse used to make the logic circuit change to its opposite state. To simplify subsequent discussion of logic circuits, only two of the infinite number of different types of waveforms possible for each type of logic circuit will be presented. This will be done by ignoring the voltage magnitude of the logic pulse, and by

considering only the relative polarity of the pulse.



THE LOGIC ONE AND ZERO

Examine the two simplified waveforms (A and B) above. Notice that unless the logic polarity or the quiescent state is specified, you can not determine if the waveforms indicate positive or negative logic.

What you can tell is that two states exist. It is this two-state principle that lays the ground-work for computation by electronic computers. Computers count pulses very rapidly to perform complex mathematical operations. Humans, however, do not have the capability to read electronic pulses. Consequently, humans are stuck with using numbers. Fortunately, a system exists that allows for the conversion of the electronic pulses of computers to the numbers in the number system used by human beings. As stated before, logic circuits have two states: the quiescent state and its opposite state. To match this, the binary number system has only two characters: a 0 and a 1. Any two distinctive voltages or current may be used to represent the logic 0 and the logic 1. For example, a negative voltage could indicate 0 and a positive voltage could indicate 1, or vice versa. Likewise, current flowing into the circuit could indicate 0, and current flowing out of the circuit could indicate 1. Similarly, a small amplitude of voltage or current could indicate 1, and a large amplitude of voltage or current could indicate 0. Pulses could also be used, with a negative pulse indicating 0 and a positive pulse indicating 1; or 0 could be indicated by the absence of a pulse, in which case the presence of a pulse could signify 1. Many combinations of logic expressions are possible; they can also be used interchangeably, since each logic element (circuit) can actually function independently as long as the desired result is achieved. Most present-day logic systems utilize polarity to define circuit state, since positive and negative voltages are easily obtained and manipulated, regardless of whether the actual logic element (circuit) employ relays, switches, diodes, or transistors. Logic circuits can be divided, according to polarity into three general classes: positive, negative, and mixed logic. As employed on a logic diagram, a signal may assume either the "active (or true)" state (logic 1), or the "inactive (or false)" state (logic 0). The signal levels used and a statement concerning whether positive or negative logic applies are usually specified explicitly on the individual logic diagram by the manufacturer or the logic designer.

In practice, many variations of logic polarity are employed; i.e., from a high positive voltage to a low positive voltage, or from a positive voltage point to ground; from a high negative voltage to a low negative voltage, or from a negative voltage point to ground; and mixed polarity, from a high positive potential to a negative (below ground) potential, and vice versa. A brief discussion of the two general classes of polarity and how each is related to logic 1 and to logic 0 is presented below.

Positive Logic

Positive logic polarity is defined as follows: When the logic 1 state has a relatively higher positive electrical level than does the logic 0 state, and the circuit is activated (operated) by the logic 1 signal, the logic polarity is considered to be positive. The following examples are typical of the manner in which positive logic may be employed:

Logic 1 = +10 volts
Logic 0 = 0 volts

Logic 1 = 0 volts
Logic 0 = -10 volts

Examine both examples. The logic 1 state is always more positive than the logic 0 state, even though the logic 0 state is negative, as in the second example above. The previous statements and definitions are particularly appropriate for d.c. switching circuits, but also apply to a.c. switching circuits as well. For example, a positive pulse can be used to simulate a positive voltage, and a negative pulse can be used to simulate a negative voltage. However, such complexity is unnecessary, since the absence of a pulse can signify the logic 0 state, and the original definition of positive polarity will still apply. That is, the logic 1 state is more positive than the no signal (or logic 0) state.

As normally used, positive logic is more adaptable to an NPN type transistor because of the NPN polarity requirements. A positive collector voltage is required to properly bias the NPN transistor; operation of the transistor produces either a low or a high positive output voltage. The use of positive logic, however, is not restricted to the NPN transistor. If the PNP transistor is connected in an a.c. system in the common-emitter (CE) configuration, the input polarity will be inverted in the output circuit. The use of NPN transistors merely makes the design of d.c. positive logic circuits easier and simpler; thus, normally, positive logic is associated with NPN transistors. At present, however, it is unimportant which type of logic polarity (positive or negative) is used, since logic components and circuits are available for all types. In fact, if a symbol such as H for the high or active state (1), and L for the low or inactive state (0) are used, logic design may be completed and circuitry devised without concern for the polarities or levels used. Once the logic design is completed, standard circuits of the proper type and polarity for the components and level to be used are selected, and the unit is constructed.

Negative Logic

Negative logic polarity is defined as follows: When the logic 1 state has a relatively more negative electrical level than the logic 0 state, and the circuit is activated (operated) by the logic 1 signal, the logic polarity is considered to be negative. The following examples are typical of the manner in which negative logic can be employed.

Logic 1 = 0 volts
Logic 0 = +10 volts

Logic 1 = -10 volts
Logic 0 = 0 volts

In both examples the logic 1 state is always more negative than the logic 0 state, even though both states are in the positive region, as in the first example above. While the above definitions of logic polarity are particularly applicable to d.c. switching circuits, they apply to a.c. circuits as well. For example, a negative pulse can be used to simulate a negative voltage, and a positive pulse can be used to simulate a positive voltage. Such a complexity is unnecessary, however, since the absence of a pulse can be assumed to indicate the logic 0 state. Thus, only a negative pulse is necessary, and the above definitions will still apply.

As normally used, negative logic is more adaptable to the PNP type transistor because of the PNP polarity requirements (see figure 3-4(B)). A negative collector voltage is required to properly bias the PNP transistor; operation of the transistor produces either a low or a high negative output voltage.

Negative logic is not limited to the PNP transistor, since use of the NPN transistor in the common emitter configuration (in an a.c. coupled circuit) can invert the polarity of the input signal and provide a negative output. The use of PNP transistors merely makes the design of d.c. negative logic circuits easier and simpler. Thus, negative logic is normally associated with PNP transistors and positive logic with NPN transistors (see figure 3-4(A)). Since logic circuits and components are available for all types of polarity, there is no particular reason why negative logic should be used in preference to positive logic. In fact, for design reasons some special computers use both (mixed) positive and negative polarities. The usual practice is to design the logic without regard to polarity or to voltage levels. Once designed, the proper type of

polarity and voltage levels for the standard logic circuit and components to be used are selected, and the unit is constructed.

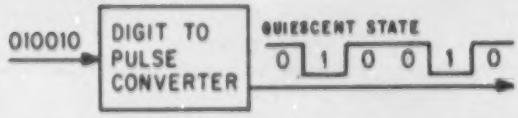
Counting Logic Pulses

At this point, it is only necessary for you to understand that a logic 1 or a logic 0 at the input of a logic circuit can cause a switching action of the switch element. Of course, which action occurs depends upon the existing state or condition of the circuit.

Refer to figure 3-5(A). These positive-going pulses at the input of a positive logic circuit would cause the switching element to change from a 0 state to a 1 state. In the lower waveform of figure 3-5(A) notice two 0's side by side. They exist because the pulse is twice the width of the 1 pulse on either side of it. Therefore, the greater width represents two pulses.

The waveform in figure 3-5(B) illustrates the same principle, only negative logic is used. That is, negative going 1's would cause the switch to change from a quiescent state to an active state.

The preceding information can be very useful to you. For instance, if the logic number input to a logic circuit and the type of logic circuit are known, you can determine both the input and output waveforms. For example, an input of 010010 in negative logic results in the below output waveform:



The input waveform is the same as the output waveform. Actually, the output waveform is determined by the type of logic circuit used.

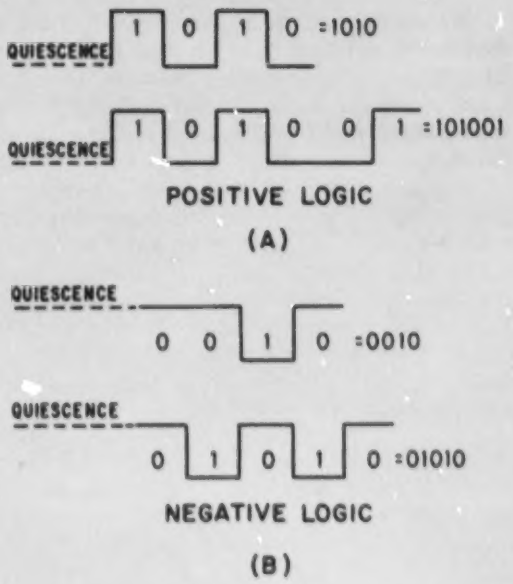


Figure 3-5.—Logic Number Configuration.

The remainder of this topic concerns the differences in the six basic logic circuits and their outputs. However, before you go on, be sure you understand what has just been covered by answering the following questions:

Q1. What is the input waveform for 1011 in positive logic?

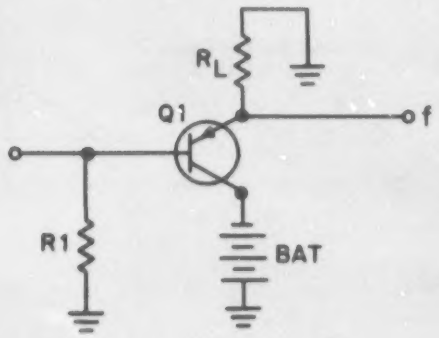
Q2. What is the negative logic input waveform for Q1?

Q3. The waveform below is positive logic. What logic number does it represent?

Q4. What logic number does the above waveform represent in negative logic?

The next problem is a little more difficult. Review its solution step-by-step, then answer Q5.

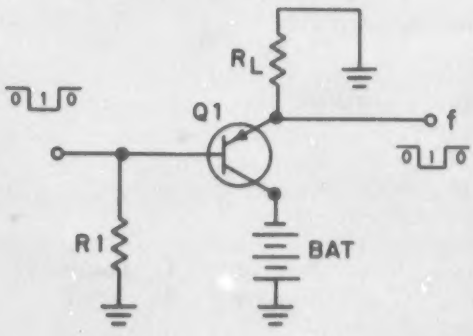
What is the logic polarity of the switch shown below?



Notice that the above logic circuit does NOT contain a lamp as did the logic circuits in figures 3-1 through 3-4. However, problem solution is the same. Now follow the steps listed below:

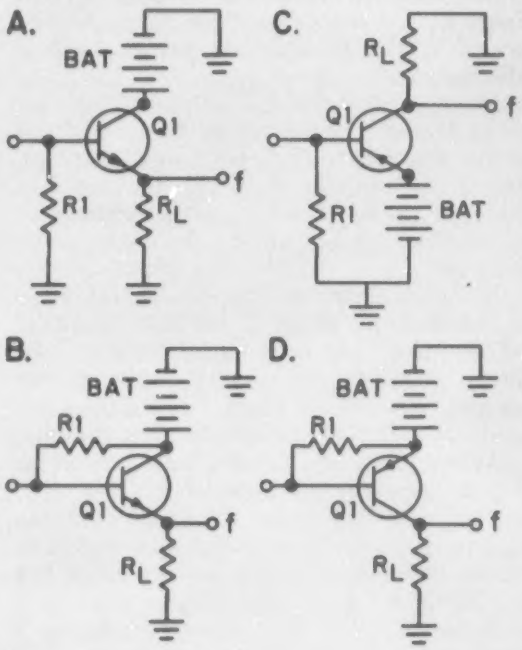
First, determine the quiescent state of the transistor. To do this, assume the input is logic 0, then determine if the transistor is biased into cutoff or saturation. In transistor Q1 the base is N-type material and is connected to the + end of the battery. The transistor is therefore cutoff and the output potential at point f is essentially at ground potential.

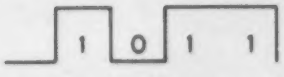
Next, determine the polarity of the input that will cause the switch element to change its state. To do this, note whether the positive bias voltage applied to the N-type material caused the transistor to cut off. If it did, a negative voltage will cause the transistor to change state, and the logic is negative, as is shown by the waveforms in the illustration below.

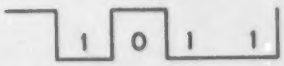


You can use this procedure to determine the logic polarity of any of the basic logic circuits with the exception of the NOT inverter. The NOT inverter will be explained later.

Q5. What polarity is the logic for each circuit below?



A1. 

A2. 

A3. 10101

A4. 01010

A5. A. Positive C. Positive
B. Negative D. Negative

THE DIODE SWITCH

In each of the logic circuits so far, a transistor is used as the switching element; however, the transistor is not the only electronic device capable of the extremely rapid switching speeds required in electronic computers. Because of their switching-speed characteristics, semiconductor diodes are also used in logic circuits. For the technician they do have a drawback: you must first know the polarity of logic in order to figure out whether the circuit is outputting a 1 or a 0.

Examine figure 3-6. The polarity of logic for the diode switch is not as easy to figure out as for the transistor switch previously discussed. This is because the bias for the diode is developed by the circuit that inputs pulses to the diode switch and not by the diode switch circuit itself.

Notice in figure 3-6 that it is impossible to tell whether the diode is forward biased or reverse biased. And, without knowing the bias voltage polarity, you cannot determine the quiescent state of the diode. Even if the input waveform were shown on the diagram, you would not know which pulse would cause the diode to switch states. However, if you look at both the diode switch and the circuit supplying input pulses to the diode switch, you should be able to determine both the quiescent state and the polarity of logic of the diode.

Refer to figure 3-7. The transistor, in a quiescent state, is conducting. Therefore, the

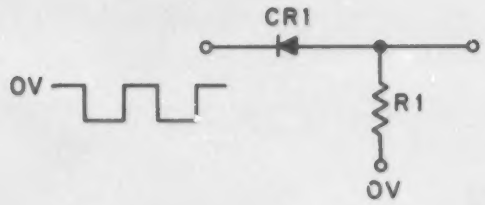


Figure 3-6.—The Diode Switch.

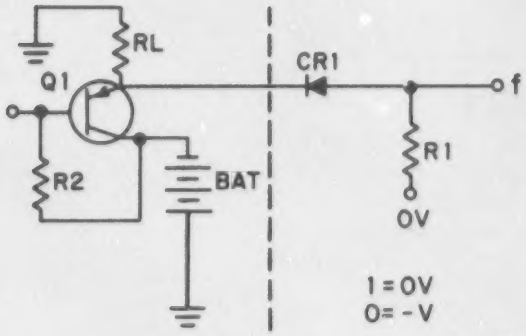
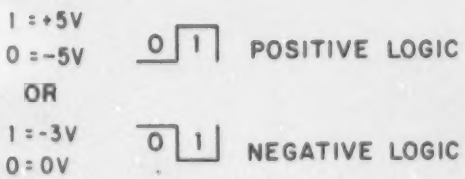


Figure 3-7.—The Diode Switch with an Input Circuit.

input voltage to the diode switch is negative, causing the diode to be forward biased. A positive 1 input to the transistor will cause it to cut off. This will reverse bias CR1, causing the output to go to 0 volts or a logic 1.

To ensure you understand the following information on diode switches, the polarity of logic will be indicated. For example,

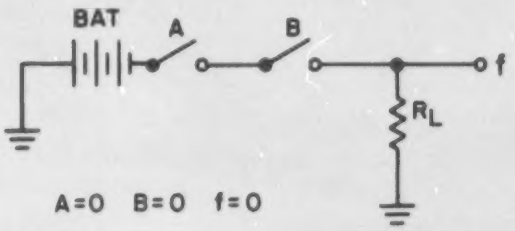


THE AND GATE

The AND gate is a logic switch that has multiple inputs (two or more). In addition, it is

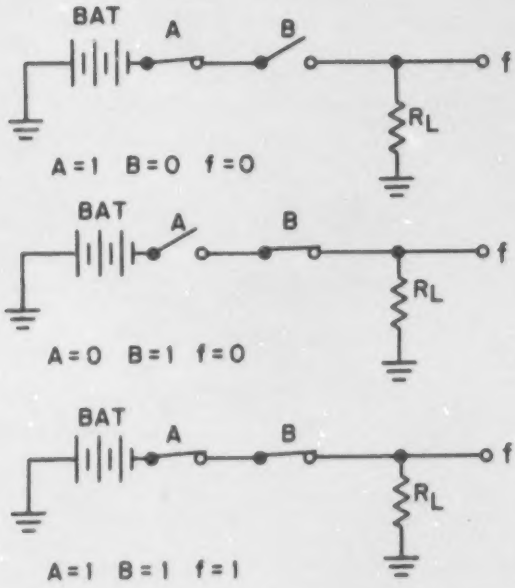
so configured that all inputs must be 1's to get a 1 output.

For an explanation of the AND gate, refer to the following diagram and read below it. Assume that switches A and B are manually operated.



Switches A and B must both be closed to get the negative voltage, or 1, to the output side, f, of the AND gate. If only A or only B is closed, the circuit is still open and the output remains at 0 volts.

This is shown in the following diagrams.

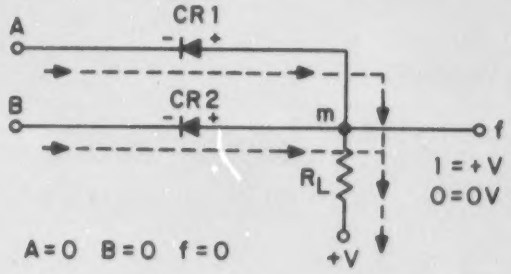


The input conditions of any logic gate and the corresponding output of the logic gate can be summarized in a truth table. The truth table for the above illustrated AND gates is shown in figure 3-8. To this point, manual switches were used in the example AND gates, but since you are studying electronics, electronic switches will now be used. As stated before, the logic polarity will be specified.

The Positive Logic Diode AND Gate

The positive logic AND gate is shown in figure 3-9.

Refer to the following illustration. Current is indicated by dashed lines and arrows. At quiescence, all 0's are inputted to the AND gate, and current flows across forward-biased diodes CR1 and CR2, through R_L to the positive source. Point m at the top of R_L is at zero-volt potential, and the output at f is a logic 0.



When a logic 1 is applied to input A and a logic 0 is applied to input B, CR1 is reverse biased. The potential difference between the cathode and anode of CR2 provides forward bias

INPUTS		OUTPUTS
A	B	f
0	0	0
0	1	0
1	0	0
1	1	1

Figure 3-8.—AND Gate Truth Table.

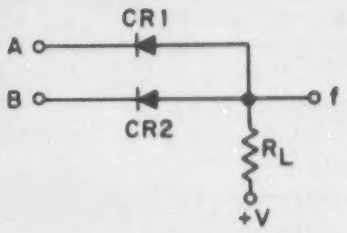
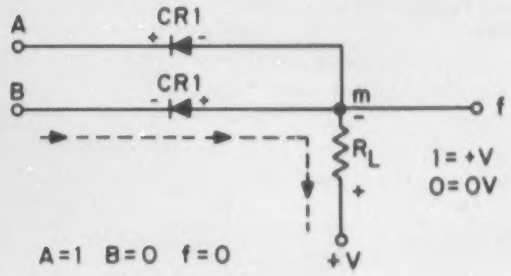
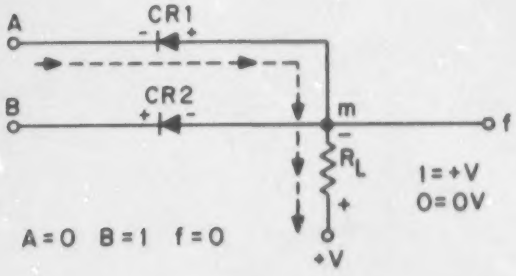


Figure 3-9.—The Positive Logic AND Gate.

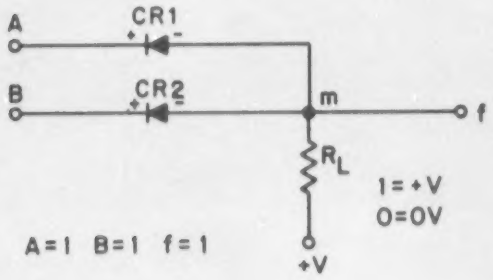
to diode CR2. Current flows through CR2 in the B input circuit and through R_L to the positive source (+V). Zero volts exists at point m and the output at f is a logic 0.



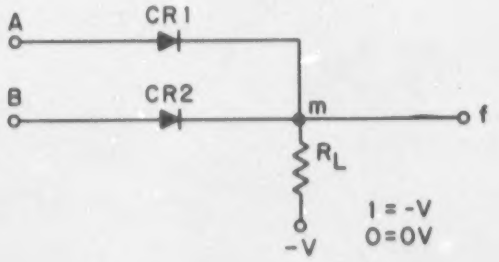
When input B is a logic 1 and A is a logic 0, the same condition exists, except CR1 is forward biased and CR2 is reverse biased. The output is a logic 0. These conditions are shown below.



When input A and input B are each a logic 1, CR1 and CR2 are both reverse biased and no current flows through R_L. Thus, point m is positive, and the output (f) is a logic 1. This is shown below.



These conditions are true for all AND gates whether they have two inputs or a hundred inputs. The negative logic AND gate works the same way, only you must first turn the diodes around, as well as reverse the logic and the source voltage. By making these changes to the diode AND gate of figure 3-9, you get the negative logic AND gate illustrated and explained below.



When A and B are 0 they are at 0-volt potential, both CR1 and CR2 are conducting, current flows through R_L, and the IR drop across R_L causes point m to be at 0-volt potential. Therefore, f is at 0-volts and is equal to a logic 0. Applying a 1 (or -V) to CR1 only or to CR2 only will allow the other diode to

conduct, keeping f at 0-volt potential and at a logic 0.

If both A and B become 1, then diodes CR1 and CR2 will be reverse biased, allowing f to go negative. At this time the output is a logic 1.

A diode gate has an undesirable electronic characteristic: it has no gain. When a series of diode logic gates feed one another, their combined circuit losses may become so great that the signal may be attenuated too much to be of use. Where gain is an important factor, transistor AND gates should be used.

THE TRANSISTOR AND GATE

The simplest way to construct a transistor AND gate is to combine the diode AND gate with the transistor switch used earlier in this topic. An example transistor AND gate is shown in figure 3-10. Refer to it as you read on. At quiescence, inputs A and B are at a logic 0 (or -V), and CR1 and CR2 are forward biased. The top of R1 (point p) is at -V. The -V is applied to the base of Q1, causing it to conduct. The top of R_L (point q) is negative (or -V), and the output at f is a logic 0. When A and B are both a logic 1 (or +V), this positive voltage cuts off both diodes, allowing point p to go to +V. Since +V is applied to the base of Q1, it causes Q1 to cut off. The top of R_L (point q) goes positive, outputting a logic 1 at f.

Examine the three-input transistor AND gate shown in figure 3-11. Note that it contains resistors instead of diodes at the input. In this circuit, Q1 is held at cutoff when all logic 0's are inputted (negative potentials). Under these conditions, no collector current can flow and the gate's output is equal to the collector supply voltage (or a logic 0). A logic 1 (or positive voltage) applied to any one or two of the inputs (A, B, C) causes current to flow through R4, decreasing the negative potential at point n. However, the voltage drop across R4 is not great enough to bring Q1 out of cutoff. When all inputs (A, B, and C) have a positive potential (+V) applied to them, the negative potential at the top of R4 (point n) is sufficiently reduced by the IR drop across R4 to allow Q1 to go into saturation. At saturation, the voltage at the top of R5 (point t) drops to near 0 volts, thus outputting a logic 1 at f.

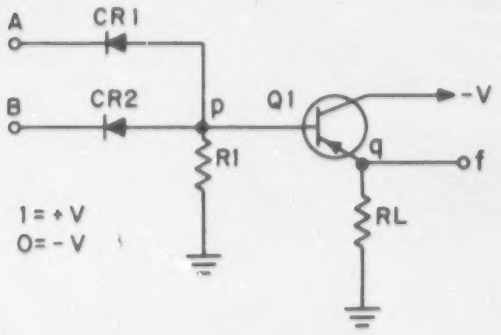


Figure 3-10.—The Transistor AND Gate.

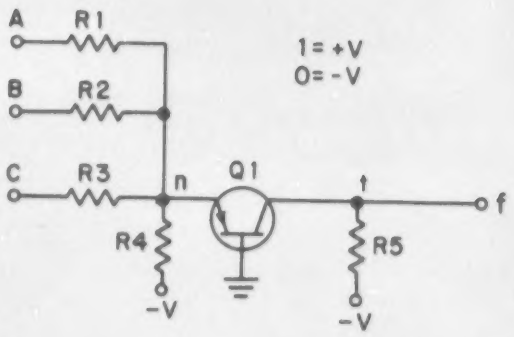
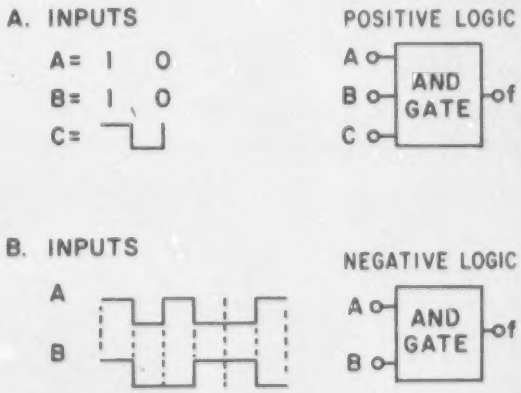
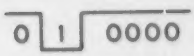


Figure 3-11.—A Three-Input Transistor AND Gate.

Q6. Draw the output waveform and logic number output for each of the following circuits.



A6. A.

B. 

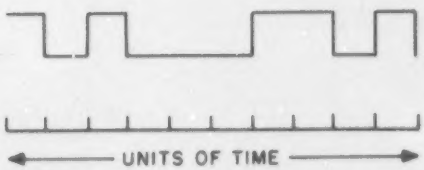
Q7. What is the output waveform for the following positive logic AND gate inputs?

A. 1 0 1 1 0
B. 0 1 1 0 1
C. 1 1 1 0 1

Q8. Draw the output waveform for the following negative logic AND gate inputs?

A. 1 1 1 1 0 1
B. 1 1 1 0 1 0
C. 0 1 1 0 1 0

Q9. What is the logic number for the following output waveform in (a) positive logic, and (b) negative logic?



Ensure that you understand the AND gate, then read the following explanation of the OR gate.

THE OR GATE

The OR gate is physically identical to the AND gate, except the OR gate gives a 1 output when ANY INPUT OR ALL INPUTS is/are 1.

Refer to the diode OR gate shown in figure 3-12. Under quiescent conditions, A and B are both logic 0's (or -V), both CR1 and CR2 are reverse biased, there is no current flowing

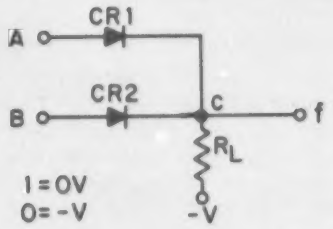


Figure 3-12.—The Diode OR Gate.

through R_L, and the top of R_L (point c) is at the negative source voltage (-V). When a 1 (0-volt potential) is applied to either A or B, the diode with the 1 applied conducts and current flows through R_L. The IR drop across R_L causes the top of R_L (point c) to go to 0 volts and the output is a logic 1. Similarly, when a 1 is applied to both inputs, the output goes to logic 1. As mentioned before, the OR gate and the AND gate are physically the same. However, if the polarity of logic to an OR gate is changed, the OR gate becomes an AND gate, as shown in figure 3-13.

Refer to figure 3-13 as you read on. Under quiescent conditions, logic 0, or 0 volts, is applied to inputs A and B, and both diodes conduct. The top of R_L (point s) goes to 0 volts because of the IR drop across the resistor. The output f then equals 0. If either A or B separately receives a logic 1 input, the associated diode cuts off, but the remaining diode continues to conduct, keeping the output at 0. Only when both inputs go to a 1 does the output become a 1.

As you can see, simply reversing the polarity of logic applied to the AND gate turns the AND

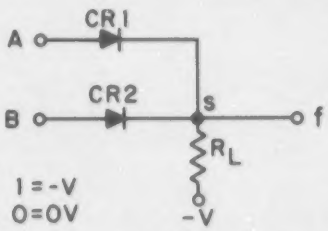
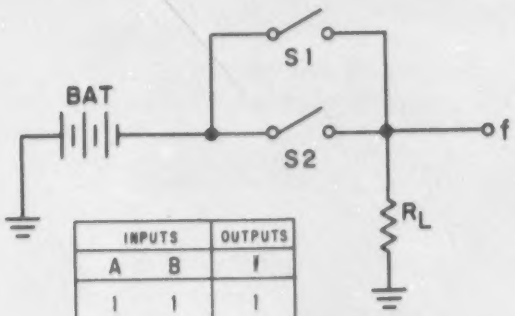


Figure 3-13.—Interchangeability of OR Gates and AND Gates.

gate into an OR gate, and vice versa. The truth table for the two-input OR gate, along with its manual switch configuration, is presented in figure 3-14.

The three-input transistor OR gate is shown in figure 3-15. In this OR gate circuit, the collector is reverse biased while the emitter is left floating. In the quiescent condition with all 0's (or -V) applied, the transistor is cut off and no current flows in the circuit. The top of R4 (point e) is at source potential (-V), and the output at f is a logic 0 (-V). With the input of a logic 1 (+V) at any of the three inputs, the emitter circuit is forward biased. Current flows through R4 and Q1. The IR drop across R4 causes point e to go to +V, causing f to output a logic 1.



INPUTS		OUTPUTS
A	B	f
1	1	1
1	0	1
0	1	1
0	0	0

Figure 3-14.—OR Gate Truth Table.

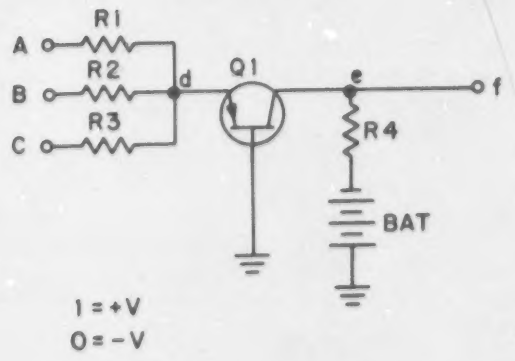
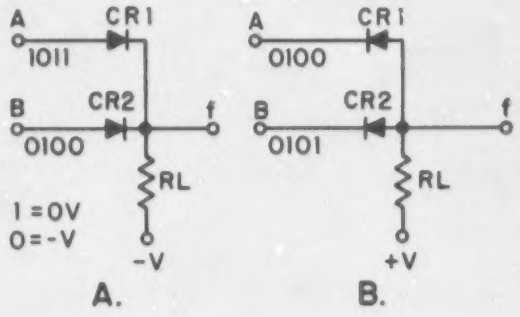


Figure 3-15.—The Three-input Transistor OR Gate.

Q10. Draw the waveform output for each of the following OR circuits.



THE NOT FUNCTION

The logic NOT function is easy to understand, since it simply inverts the input. When a logic 1 is inputted to a NOT function, it is outputted as a logic 0. A logic 0 when inputted to a NOT function, is outputted as a logic 1.

Refer to the positive NOT function shown in figure 3-16. At quiescence, with a logic 0 input (0 volts), transistor Q1 is held in cutoff by the negative voltage (-V) applied to its base. Note that -V is applied across R2 to Q1. When Q1

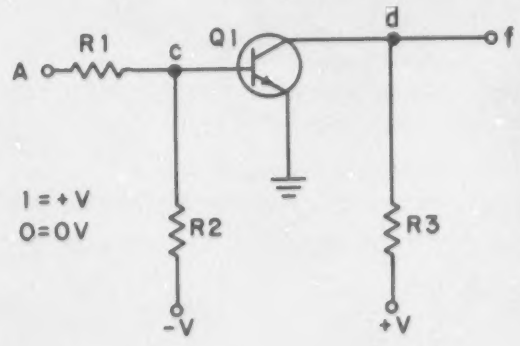
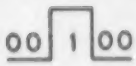
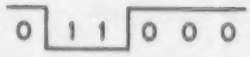
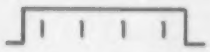


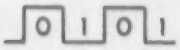
Figure 3-16.—The Positive NOT Function.

A7. 

A8. 

A9. (a) 1010001101
(b) 0101110010

A10. A. 

B. 

cuts off, the top of R3 (point d) goes to the +V source potential and the output at f is a positive voltage (or a logic 1). When a logic 1 (or +V) is applied to the input, the top of R2 (point c) becomes positive due to the IR drop across R2. Q1 goes into saturation. The saturation current flows from ground through the emitter, the base, and the collector, and then through R3 to the +V source. Because of the saturation current, Q1 is a virtual short. The top of R3 (point d) goes to ground potential, causing f to output a logic 0 (or 0 volts).

The negative NOT function is the same basic circuit as the positive NOT function, except the transistor is of the PNP type. Also, the bias voltages are reversed, as shown in figure 3-17.

Examine figure 3-17. Alternately apply a logic 1 and a logic 0 to input A, and then answer Q11, Q12, and Q13.

Q11. What is the waveform at the output if the input is 101? (Remember, this is a negative logic gate.)

Q12. If a logic 1 is applied to input A, does the transistor cutoff or go into saturation?

Q13. What is the waveform at the output when the input is 101101?

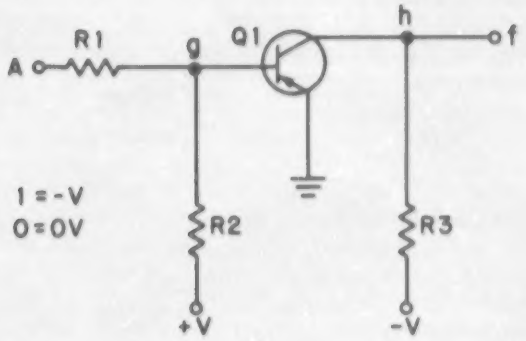


Figure 3-17.—The Negative NOT Function.

THE NAND GATE

Compare figures 3-11 and 3-18. Note that Q1 in the AND gate is common-base configured, and that Q1 in the NAND gate is common-emitter configured. The NAND gate is just an AND gate with its transistor configured so a logic input will appear inverted in the output. Actually, the transistor amplifier in a transistor NAND gate functions as an inverter.

Refer to figure 3-18 as you read the explanation of the NAND gate. With a logic 0 at any of the inputs, Q1 is cut off by the positive voltage applied from the top of R4 (point x) to the base. With the transistor cutoff, the top of R5 (point y) is at negative source potential (-V) and the output at f is a logic 1 (or -V). When a logic 1 (or -V) is applied to all three inputs, the top of R4 (point x) goes negative, causing Q1 to go into saturation. This effectively shorts the top of R5 (point y) to ground, or +V. The output, therefore, is a logic 0.

The NAND gate is the opposite of the AND gate. The AND gate requires all 1's at its inputs in order to output a 1. The only possible way not to get a 1 from the NAND gate is to apply 1's to all the inputs.

If only one input to the AND gate is a 0, the output of the AND gate is a 0. In the NAND gate, the opposite is true. If any input is a 0, the output is a 1.

In binary (two-state) logic, when a given state is the opposite of another state, it is called a complementary state. More will be said about this later.

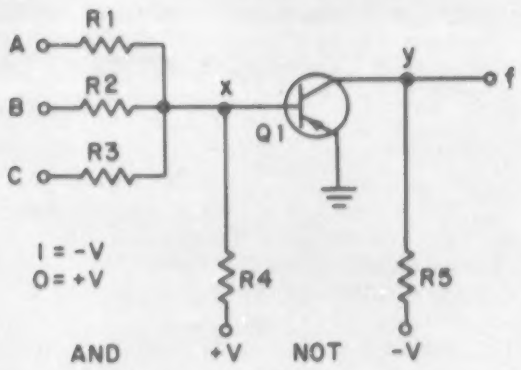


Figure 3-18.—The NAND Gate.

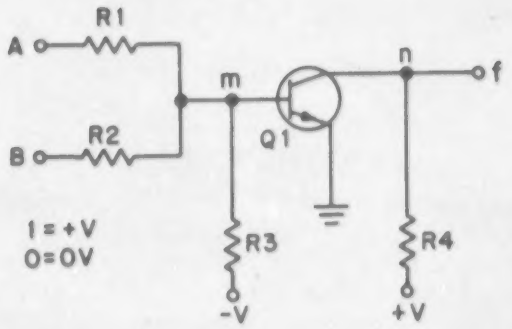


Figure 3-19.—The NOR Gate.

The truth tables for a NAND gate and an AND gate are shown below. Notice that for identical sets of input conditions in both truth tables, the output conditions are opposite. Thus, they are complementary to each other.

A	B	f
0	0	1
1	0	1
0	1	1
1	1	0

NAND

A	B	f
0	0	0
1	0	0
0	1	0
1	1	1

AND

The NOR gate is the complement of the OR gate. A 1 is outputted from the OR gate when any input is a 1. A 1 is outputted from a NOR gate only when ALL inputs are 0's. This is summarized in the following truth tables:

A	B	f
0	0	1
1	0	0
0	1	0
1	1	0

NOR

A	B	f
0	0	0
1	0	1
0	1	1
1	1	1

OR

THE NOR GATE

The NOR is a combination of an OR gate and an inverter. When the OR gate portion outputs a 1, the inverter portion converts the 1 to a 0. When the OR gate portion outputs a 0, the inverter portion outputs a 1.

Refer to the circuit in figure 3-19. With all 0's at the inputs, the base of Q1 is reverse biased by the negative voltage developed across R3. Q1 is cut off, and the top of R4 (point n) is at source potential (+V). The output at f is a logic 1. A positive voltage from a logic 1 applied to any of the inputs provides forward bias to the base of Q1, causing Q1 to go into saturation. This effectively grounds the top of R4 (point n), causing the output at f to be 0 volts, or a logic 0.

THE EXCLUSIVE OR GATE

The Exclusive OR gate is a modified OR gate. When any single input is a 1, the circuit outputs a 1. When ALL of the inputs are 1's, the circuit outputs a 0. This is indicated in the following truth tables. Compare these two tables closely.

A	B	f
0	0	0
1	0	1
0	1	1
1	1	1

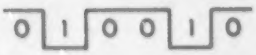
OR

A	B	f
0	0	0
1	0	1
0	1	1
1	1	0

EXCLUSIVE OR

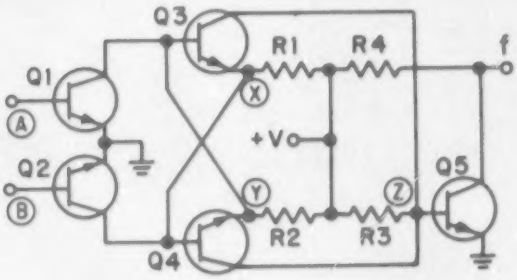
A11.

A12. SATURATION

A13. 

As you read below, refer to the schematic diagram of the Exclusive OR gate circuit in figure 3-20.

Q1 and Q2 each form a NOT circuit, while transistors Q3, Q4, and Q5, form an OR circuit. With a logic 0 (or 0 volts) applied at both inputs, Q1 and Q2 cut off. No current flows through R1 and R2. Points (X) and (Y) are near source potential (+V), thus a positive voltage is applied to the emitters of Q3 and Q4, cutting Q3 and Q4 off. With no current through Q3 and Q4, point (Z) is at +V potential. This applies a positive voltage to the base of Q5, driving Q5 into saturation and effectively placing point f at ground, or 0 volts. The output is a logic 0.



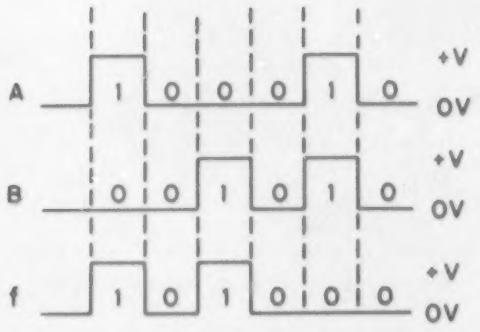


Figure 3-20.—Exclusive OR Gate.

When a logic 1 is applied to (A) and a logic 0 is applied to (B), Q1 saturates, while Q2 remains cutoff. Current flows from ground through Q1 through R2 to +V. The IR drop across R2 causes (Y) to be at ground potential. This ground is applied to the base of Q3 and to the emitter of Q4. Q3 is cutoff. Q4 saturates, causing an IR drop across R3. This causes point (Z) to be at ground potential, forcing Q5 to cutoff, thus allowing point f to go to +V (the source potential). The output is a logic 1.

When a logic 1 is applied to (B) and a logic 0 is applied to (A), circuit operation is the same, except Q2 and Q3 conduct while Q1 and Q4 remain cutoff. Notice that the current through Q3 also passes through R3. The IR drop across R3 causes point (Z) to go to ground and cut off Q5. This causes f to output a 1.

When a logic 1 is applied to both inputs, both Q1 and Q2 saturate, causing points (X) and (Y) to go to ground potential. Q3 and Q4 go into cutoff. The IR drop across R3 ceases and point (Z) goes to the +V source potential. The positive voltage applied to the base of Q5 causes the transistor to go into saturation, placing f at ground potential. A logic 0 is outputted.

Now that you have seen how the Exclusive OR gate operates, answer the following questions about the circuit in figure 3-20:

Q14. With a 1 applied to both inputs at the same time, why is the voltage at point f equal to zero volts?

Q15. With 0's applied to both inputs, are Q3 and Q4 saturated or cut off?

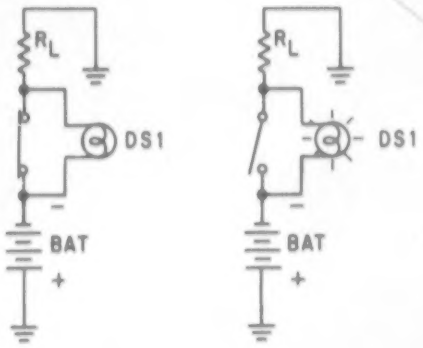
From this topic you learned the electronics behind logic gates. In the next topic, you will learn the logic behind logic gates.

SUMMARY OF LOGIC CIRCUITS

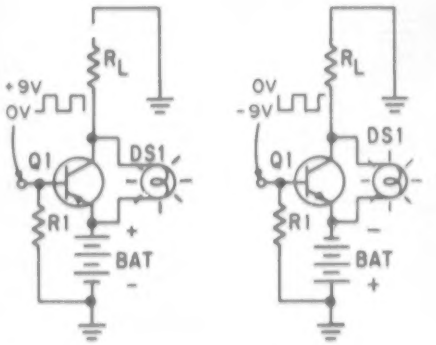
This topic gave you an introduction to the basic principles of logic as applied to computer technology. The following is a summary of the main points you covered.

LOGIC SWITCHING GATES–All computer functions are based on the operation of logic switching gates. These gates are actually electronic switches which have two states: ON or OFF.

When the gate is on, it is said to be "gated on", and when it is off, it is "gated off". When a logic gate is in either of these states, the conditions which control the gate have been met.



LOGIC POLARITY–The polarity of the input signal which causes a logic gate to change state is called its logic polarity. The polarity of the input signal can be either positive or negative. The factor which determines the polarity of a logic circuit is based on circuit design. The important idea here is not the design of the circuit, but the polarity of the logic which causes the circuit to change state. If the signal pulse that causes the circuit to change from an off condition to an on condition is positive going, then the logic used is positive logic. If a negative going pulse is used to turn the gate on, then the logic used is negative logic. Remember: Logic polarity is directly related to the polarity of the signal used to control the logic gate.



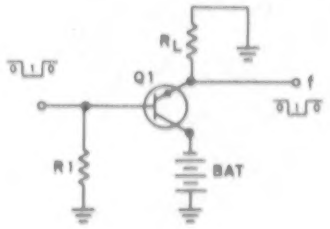
THE LAMP LIGHTS WHEN POSITIVE-GOING PULSES ARE PRESENT

POSITIVE LOGIC

THE LAMP LIGHTS WHEN NEGATIVE-GOING PULSES ARE PRESENT

NEGATIVE LOGIC

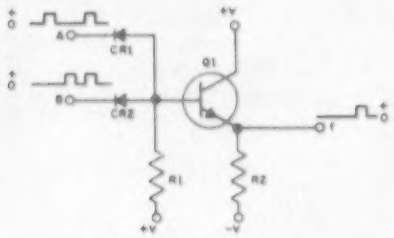
LOGIC ONE AND LOGIC ZERO--Logic circuits have only two states–ON or OFF. This two-state condition can be matched to the binary number system which uses only two digits: "1" and "0". In a logic circuit, the ON condition can be represented by a logic 1 and the OFF condition can be represented by a logic 0. Additionally, the ON condition could have been represented by a YES, a TRUE (T) or a HIGH (H) and the off condition could have been represented by a NO, a FALSE (F), or a LOW (L).



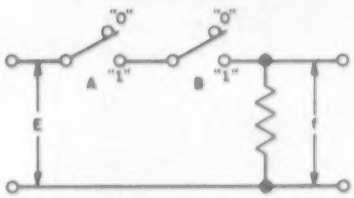
A14. **Q5 saturates, causing f to go to ground potential**

A15. **Cutoff**

THE AND GATE–An AND gate is a logic switch with two or more inputs. The AND gate is configured so that each input must be a logic 1 to get a logic 1 output. The input conditions of any logic gate and its corresponding output are summarized in a TRUTH TABLE.



A. CIRCUIT SCHEMATIC

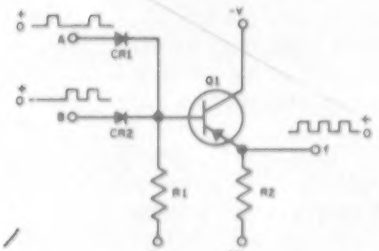


B. MECHANIZATION

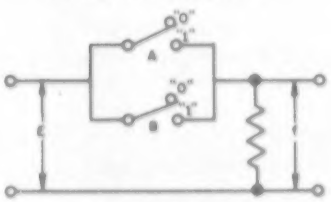
A	B	f (A,B) = AB
0	0	0
0	1	0
1	0	0
1	1	1

C. TRUTH TABLE

THE OR GATE–The OR gate is physically identical to the AND gate, except the OR gate will give a 1 output anytime there is a 1 present at any one of the inputs.



A. CIRCUIT SCHEMATIC

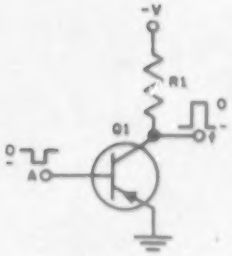


B. MECHANIZATION

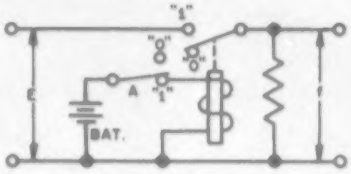
A	B	f (A,B) = A+B
0	0	0
0	1	1
1	0	1
1	1	1

C. TRUTH TABLE

THE NOT FUNCTION–The NOT function is nothing more than an inverter. If a logic 1 is inputted to an inverter it is outputted as a logic 0. Conversely, if a logic 0 is inputted to an inverter, it is outputted as a logic 1.



A. CIRCUIT SCHEMATIC

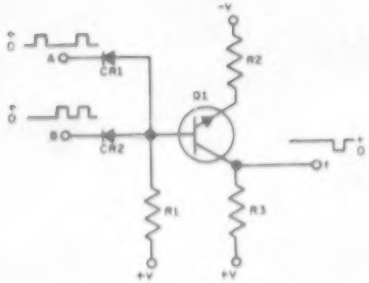


B. MECHANIZATION

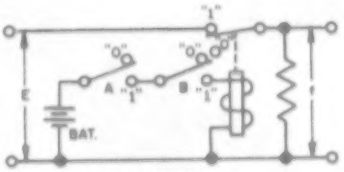
A	f (A) = $\bar{A}$
1	0
0	1

C. TRUTH TABLE

THE NAND GATE–The NAND gate is an AND gate with an inverter attached to its output. With a logic 0 at any one of the inputs, the output is a logic 1. Conversely, anytime logic 1's are applied to all inputs, the output is a logic 0. The NAND gate can be described as a complemented AND gate.



A. CIRCUIT SCHEMATIC

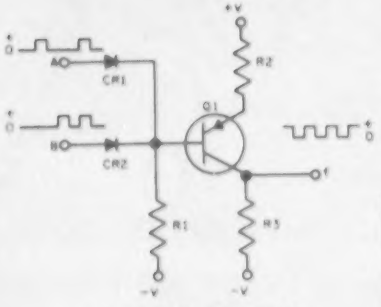


B. MECHANIZATION

A	B	AB	f (A,B) = $\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

C. TRUTH TABLE

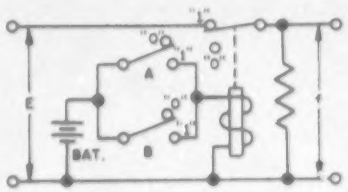
THE NOR GATE–The NOR gate is an OR gate with an inverter attached to its output. With logic 0's applied to all inputs, the output is a logic 1. When a logic 1 is applied to any one of the inputs, the output is a logic 0. The NOR gate can be described as a complemented OR gate.



A. CIRCUIT SCHEMATIC

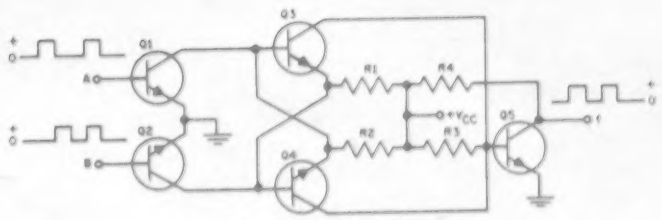
A	B	A+B	$f(A,B) = \overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

C. TRUTH TABLE

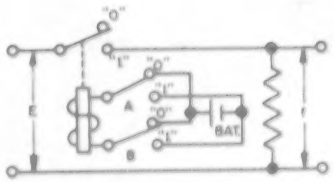


B. MECHANIZATION

THE EXCLUSIVE OR GATE–The Exclusive OR gate is an OR gate which is modified so that when any single binary 1 is applied as an input, the output is also a binary 1. When all inputs are a 1, the output is a binary 0.



A. CIRCUIT SCHEMATIC



B. MECHANIZATION

A	B	$f(A,B) = A\bar{B} + \bar{A}B$
0	0	0
0	1	1
1	0	1
1	1	0

C. TRUTH TABLE

APPENDIX I

GLOSSARY OF TERMS

ABSORPTION, LAW OF–In Boolean algebra, the law which states that the odd term will be absorbed when a term is combined by logical multiplication with the logical sum of that term and another term, or when a term is combined by logical addition with the logical product of one term and another term (e.g., $A(A + B) = A$ or $A + AB = A$).

AND CIRCUIT–See AND GATE.

AND DEVICE–A device whose output is in the logic 1 state if and only if all the input signals are in the logic 1 state.

AND GATE–(1) An electronic gate whose output is energized only when every input is in its prescribed state. An AND gate performs the function of the logical "AND". Also called an AND circuit.

(2) A binary circuit, with two or more inputs and a single output, in which the output is a logic 1 ONLY when all inputs are a logic 1, and the output is a logic 0 when any one of the inputs is a logic 0.

ANODE–(1) A positive electrode; the plate of a vacuum tube; the electrode of an electron tube through which a principle stream of electrons leaves the interelectrode space.

(2) The positive electrode of an electrochemical device, such as a primary or secondary cell, toward which the negative ions are drawn.

(3) The semiconductor-diode terminal that is positive with respect to the other terminal when the diode is biased in the forward direction.

BASE–(1) A reference value.

(2) A number that is multiplied by itself as many times as indicated by an exponent.

(3) Same as radix.

(4) The region between the emitter and collector of a transistor which receives minority carriers injected from the emitter. It is the element which corresponds to the control grid of an electron tube.

BIAS–(1) Vacuum tube: the difference of potential between the control grid and the cathode.

(2) Transistor: the difference of potential between the base and emitter and the base and collector.

(3) Magnetic amplifier: the level of flux density in the magnetic amplifier core-under no-signal condition.

BIAS CURRENT–Current which flows through the base-emitter junction of a transistor and is adjusted to set the operating point of the transistor.

BINARY–(1) A number system that uses a base, or radix, of 2. There are two digits (1) and (0) in the binary system.

(2) Pertaining to a characteristic which involves the selection, choice, or condition in which there are at most two possibilities.

(3) A bistable multivibrator (Flip-Flop) is one example of a binary device.

BINARY CODE–A method of representing two possible conditions (on or off, high or low, one or zero, the presence of a signal or absence of a signal). Electronic circuits designed to work such that there are only two possible conditions.

BINARY-CODED–The state in which conditions are expressed by a series of binary digits (0's and 1's).

BINARY DIGIT–(1) A character that represents one of the two digits in the number system that has a radix of two.

(2) Either of the digits, 0 or 1, which may be used to represent the binary conditions of on or off.

BINARY NOTATION–See BINARY NUMBER SYSTEM.

BINARY NUMBER SYSTEM–A number system using two digits, symbols, or characters (usually 1 and 0).

BINARY POINT–The radix point which separates powers of two and fractional powers of two in a binary number.

BINARY SYSTEM–A number system which consists of two numbers i.e. 1 and 0.

BISTABLE–A device which is capable of assuming either one of two stable states.

BISTABLE MULTIVIBRATOR–See FLIP-FLOP.

BOOLEAN–(1) Pertaining to the process used in the algebra formulated by George Boole.

(2) Pertaining to the operations of formal logic.

BOOLEAN ALGEBRA–A system of logic dealing with on-off circuit elements associated by such operators as AND, OR, NAND, NOR, and the NOT function.

BOOLEAN LOGIC–See BOOLEAN ALGEBRA.

CARRY–(1) One or more digits, produced in connection with an arithmetic operation that is/are forwarded to another digit place for processing there.

(2) The number represented by the digit or digits in (1) above.

CHARACTER–A letter, digit, or other symbol that is used as part of the organization, control, or representation of information.

CLASS C OPERATION–Operation of a transistor or vacuum tube with bias considerably beyond cut-off so that collector or plate current flows for less than one-half cycle.

COMMON IDENTITIES, LAW–In Boolean algebra this law states that anytime the expression $A(\bar{A}+B)=AB$ or $\bar{A}+AB=A+B$ appears, it can immediately be simplified to AB without going through the process of using the distributive law, complementary law, or the law of union to simplify.

COMMUTATIVE LAW–In Boolean algebra this law states that changing the order of the terms in an equation will not affect the value of the equation. Example: $A + B = B + A$; $A \cdot B = B \cdot A$

COMPLEMENT–A number or state that is the opposite of a specified number or state. The negative of a number is often represented by its complement.

COMPLEMENTARY LAW–In Boolean algebra this law states that the logical addition of a quantity and its complement will result in 1 and the logical multiplication of a quantity and its complement will result in a product of 0.

COMPLEMENT NUMBER–A number which, when added to another number, gives a sum equal to the base of the number system of operation. For example, in the decimal number system, the complement of 1 is 9.

COMPUTER–A data processor that can perform substantial computation, including numerous arithmetic or logic operations, without intervention by a human operator during the run.

CONCURRENT–Pertaining to the occurrence of two or more events or activities within the same specified interval of time.

CUTOFF–(1) The condition which occurs when the emitter-base junction of a transistor has zero bias or is reverse biased and cuts off or stops the flow of collector current.

(2) The minimum level of bias which cuts off, or stops, the flow of plate current in an electron tube.

(3) The frequency above or below which a frequency-selective or frequency-sensitive circuit fails to respond.

DATA–(1) A representation of facts, information, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by human or automatic means.

(2) Any representations such as characters to which meaning is or might be assigned.

DATA PROCESSING–The execution of a systematic sequence of operations performed upon data. Synonymous with information processing.

DECIMAL–Pertaining to the number representation system with a radix of ten.

DECIMAL DIGIT–In decimal notation, one of the characters 0 through 9.

DECIMAL NOTATION–A fixed radix notation where the radix is ten.

DECIMAL NUMERAL–A decimal representation of a number.

DECIMAL POINT–The radix point in decimal representation.

DeMORGAN's THEOREM–A theorem which states that the inversion of a series of AND applications is equal to the same series of inverted OR applications, or the inversion of a series of OR applications is equal to the same series of inverted AND applications. In symbols, $\overline{A \cdot B \cdot C} = \overline{A} + \overline{B} + \overline{C}$ or $\overline{A + B + C} = \overline{A} \cdot \overline{B} \cdot \overline{C}$

DIGIT–A symbol that represents one of the nonnegative integers smaller than the radix. For example, in decimal notation a digit is one of the characters from 0 through 9.

DIGITAL COMPUTER–(1) A computer in which discrete representation of data is used.

(2) A computer that operates on discrete data by performing arithmetic and logic processes on these data.

DIODE–(1) Vacuum tube–a two element tube that contains a cathode and plate.

(2) Semiconductor–a material of either germanium or silicon that is manufactured to allow current to flow in only one direction. Diodes are used as rectifiers and detectors.

DISTRIBUTIVE LAW–In Boolean algebra the law which states that if a group of terms connected by like operators contains the same variable, the variable may be removed from the terms and associated with them by the appropriate sign of operation (e.g. A(B+C)=AB+AC).

DOUBLE NEGATIVE LAW–In Boolean algebra, the law which states that the complement of a complement is the equivalent of the original term.

ELECTRONIC SWITCH–A circuit which causes a start-and-stop switching action by electronic means.

EMITTER–(1) The electrode within a transistor from which carriers are usually emitted.

(2) In a vacuum tube, the cathode.

EXCLUSIVE OR–A function whose output is a 1 if one and only one of the input variables is a 1.

EXCLUSIVE OR GATE–A gate that produces a logic 1 output when the inputs are different, but not when they are the same.

EXPONENT–The numeral written in superscript (10^2) which indicates the power to which the base is to be raised.

EXPRESSION–A validated series of variables, constants, and functions that can be connected by operating symbols to describe a desired computation.

FACTOR–Any of the elements, quantities, or symbols which, when multiplied together, form a product.

FIXED BIAS–A bias voltage of constant value, such as one obtained from a battery, a power supply, or a generator.

FLIP-FLOP–A device having two stable states and two input terminals (or types of input signals), each of which corresponds with one of the two states. The circuit remains in either state until caused to change to the other state by application of a voltage pulse. A similar bistable device with an input which allows it to act as a single-stage binary counter.

FORWARD BIAS–A bias voltage applied to a semiconductor junction with polarity such that relatively high current flows through the junction.

FUNCTION–A specific purpose of an entity, or its characteristic action.

GATE–As applied to logic circuitry, one of several different types of electronics devices that will provide a particular output when specified input conditions are satisfied. Also, a circuit in which a signal switches another signal on or off.

GATING–The process of selecting those portions of a wave which exist during one or more selected time intervals or which have magnitudes between selected limits. Also, the application of a specific waveform to perform electronic switching.

GROUND–The point in a circuit used as a common reference point for measuring purposes. Also, to connect some point of an electrical circuit or some item of electrical equipment to earth or the conducting medium used in lieu thereof.

HEXADECIMAL–Same as SEXADECIMAL. (1) A number system with a base of sixteen, also pertains to conditions, choice, or selection that has sixteen possible values or states.

(2) Pertaining to a characteristic or property involving a selection, choice, or condition in which there are sixteen possibilities.

HEXADECIMAL SYSTEM–Pertaining to the number system with a radix of sixteen. It uses the ten digits of the decimal system and the first six letters of the English alphabet.

IDENTITY, LAW OF–In Boolean algebra, the law which states that any expression is equal to itself (e.g., $A = A$, or $\overline{A} = \overline{A}$).

INDEMPOTENT LAW–In Boolean algebra, combining a quantity with itself either by logical addition or logical multiplication will result in a logical sum or product that is the equivalent of the quantity (e.g., $A + A = A$; $A \cdot A = A$).

INPUT–One, or a sequence of, input state(s). Also the current, voltage, power, or driving force applied to a circuit or device.

INPUT/OUTPUT–Pertaining to either input or output or both.

INTERSECTION, LAW OF–In Boolean algebra, the law which states that if one input to an AND gate is already TRUE, then the output will depend upon the state of the other inputs only.

INVERT–To change a physical or logical state to its opposite.

INVERTER–A circuit with one input and one output. Its function is to invert or reverse the input. When the input is high, the output is low, and vice versa. The inverter is sometimes called a NOT circuit, since it produces the reverse of the input.

LEAST SIGNIFICANT DIGIT (LSD)–The LSD is the digit whose position within a given number expression has the least weighting power.

LOAD–The power that is being delivered by any power producing device. The equipment that uses the power from the power producing device.

LOGIC–The basic principles and applications of truth tables, interconnections of off-on circuit elements, and other factors involved in mathematical computation in automatic data processing systems and other devices.

LOGIC CIRCUIT–The primary control information processing in a digital equipment; made up of electronic gates and named because their operation is described by simple equations of a specialized logic algebra.

LOGIC DIAGRAM–In computers and data processing equipment, a diagram representing the logical elements and their interconnections without necessarily expressing construction or engineering details.

LOGIC ELEMENT–The smallest building blocks which can be represented by operators in an appropriate system of symbolic logic. Typical logic elements are the AND-gate and the flip-flop, which can be represented as operators in a suitable symbolic logic. Also a device that performs the logic function.

LOGIC INSTRUCTION–Any instruction that executes a logic operation that is defined in symbolic logic, such as AND, OR, NAND, or NOR.

LOGIC OPERATION–A nonarithmetical operation in a computer, such as comparing, selecting, making references, matching, sorting, and merging, where the logical YES or NO quantities are involved.

LOGIC SWITCH–A diode matrix (See MATRIX) or other switching arrangement that is capable of directing an input signal to one of several outputs.

LOGIC SYMBOL–A symbol used to represent a logic element graphically. Also a symbol used to represent a logic operator.

LSD–See Least Significant Digit.

MATRIX–In computers, a logic network in the form of an array of input leads and output leads with logic elements connected at some of their intersections.

MECHANIZATION–Using electric or electromechanical switches to represent logic circuits (AND, OR, NOT, NOR, NAND).

MOST SIGNIFICANT DIGIT (MSD)–The MSD is the digit whose position within a given number expression has the greatest weighting power.

MSD–See Most Significant Digit.

MULTIVIBRATOR–A type of relaxation oscillator (containing two transistors) that generates nonsinusoidal waves. The output of each of its two transistors is coupled to the input of the other to sustain oscillations.

NAND–A logic function of A and B which is true if either A or B is false.

NAND CIRCUIT–A combination of a NOT function and an AND function in a binary circuit that has two or more inputs and one output. The output is logic 0 only if ALL inputs are logic 1; it is logic 1 if ANY input is logic 0.

NEGATION–The process of inverting the value of a function or variable.

NEGATIVE LOGIC–The form of logic in which the more positive voltage level represents a logic 0, FALSE, or LOW and the more negative voltage represents a logic 1, TRUE, or HIGH.

NOR–A logic function of A and B that is true if both A and B are false.

NOR device–(1) A device that has its output in the logic 1 state if and only if ALL of the input signals assume the logic 0 state.

NOR GATE–An OR gate which is followed by an inverter to form a binary circuit in which the output is a logic 0 if ANY of the inputs is a logic 1 and is a logic 1 only if ALL the inputs are a logic 0.

NOT CIRCUIT–A binary circuit with a single output that is always the opposite of the input. Also called an inverter circuit.

NOT DEVICE–A device which has its output in the logic 1 state if and only if the input signal assumes a logic 0 state.

NOT FUNCTION–A circuit which inverts the input signal. (Same as NOT DEVICE or CIRCUIT).

NUMBER–(1) A mathematical entity that may indicate quantity or amount of units.

(2) Loosely, a numeral. An abstract mathematical symbol for expressing a quantity. In this sense, the manner of representing the number is immaterial. Take 26, for example. This is its decimal form–but it could be expressed as a binary (base 2), octal (base 8), or hexadecimal (base 16) number.

NUMBER REPRESENTATION–The representation of numbers by agreed sets of symbols according to agreed rules.

NUMBER REPRESENTATION SYSTEM–An agreed set of symbols and rules for number representation.

NUMBER SYSTEM–Loosely, a number representation system. Any system for the representation of numbers (See POSITIONAL NOTATION).

NUMERAL–(1) A discrete representation of a number. For example, twelve, 12, XII, 1100_2 are four different numerals that represent the same number.

(2) A numeric word that represents a number.

OCTAL NUMBER SYSTEM–A number system which is based on powers of eight. This system is used extensively in computer work.

OR CIRCUIT–See OR Gate.

OR DEVICE–A device the output of which is a logic zero if and only if ALL the input signals are logic zeros.

OR GATE–A gate that performs the logic OR function. It produces an output whenever any one or more of its inputs is/are energized.

POLARITY–The characteristic of having magnetic poles of electric charges.

POSITIONAL NOTATION–A numbering system in which a number is represented by means of a stated set of symbols or digits, such that the value contributed by each symbol or digit depends upon its position as well as upon its value.

POSITIONAL WEIGHTING–The value given a digit based on the digit's position within a given number.

POSITIVE LOGIC–The form of logic in which the more positive logic level represents 1 and the more negative level represents 0.

QUIESCENCE–(1) The state of an amplifier with no signal applied.

(2) The operating conditions that exist in a circuit when no input signal is applied to the circuit.

QUIESCENT–The condition of a circuit when no input signal is being applied to it.

QUIESCENT STATE–The period during which a transistor, tube, or other circuit element is not performing an active function in the circuit.

RADIX–Also called the base. The number of distinct symbols used in a number system. For example, since the decimal number system uses ten symbols i.e., (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), the radix is 10. In the binary number system the radix is 2, because there are only two symbols i.e., (0, 1).

RADIX POINT–Also called binary point, octal point, decimal point, etc., depending on the number system.

REVERSE BIAS–An external voltage applied to a diode or semiconductor junction to reduce the flow of electrons across the junction. (Also called back bias).

SATURATED–The operating point of a transistor, diode, or electron tube in which there is no further increase in output current or voltage when the base or cathode current/voltage is increased.

SATURATION–The condition existing in any circuit when an increase in the input signal produces no further change in the output.

SEXADECIMAL–Same as hexadecimal.

STATIC–A fixed nonvarying condition, without motion.

TRUTH TABLE–A table that describes a logic function by listing all possible combinations of input values and indicating, for each combination, the true output values.

UNIT–A single object or thing.

VARIABLE–A representative symbol that can assume any of a given set of values.

VEITCH DIAGRAM–Diagrams consisting of joined squares which are used to give a graphic representation of basic logic relations.

VINCULA–Plural of Vinculum (See below).

VINCULUM–A straight horizontal line placed over one or more members of a compound logic expression to negate or complement. Additionally, used to join two or more members together.

INDEX

A

AND gate, Boolean Algebra, 20
AND gate, logic polarity, 78-81
  positive logic diode AND gate, 79-81
Associative Law, Boolean Algebra, 31-33

B

Binary system, 3
Boolean Algebra, 20-70
  AND gate, 20
  Boolean laws and theorems, 30-48
    Associative Law, 31-33
    Commutative Law, 30, 31
    Complementary Law, 34-36
    DeMorgan's Theorem, 37-43
    Distributive Law, 43-45
    Double Negative Law, 33, 34
    Idempotent Law, 33
    Law of Absorption, 45-47
    Law of Common Identities, 47, 48
    Law of Identity, 30
    Law of Intersection, 36
    Law of Union, 36
  input inverter, 24-30
  NAND gate, 23
  NOR gate, 23
  NOT function, 23
  OR gate, 21-23
  summary of Boolean Algebra, 62-70
  summary of Boolean laws, 48-50
  Veitch diagram, 50-62
    summary of the four-variable Veitch diagram, 60
    summary of three-variable Veitch diagram, 58-60
    thirty-two-square Veitch diagram, 61

C

Commutative Law, Boolean Algebra, 30, 31
Complementary Law, Boolean Algebra, 34-36
Computers, number systems, 1-4
  decimal, binary, octal and hexadecimal number systems, 3, 4
  historical background, 1
  one-to-one counting, 2
Conversion between hexadecimal and binary, 16, 17
Conversion from decimal to octal and binary, 11, 12
Conversion to binary from octal, 10
Conversion to decimal, number systems, 7-9
Conversion to octal from binary, 9, 10
Counting logic pulses, 76, 77

D

Decimal system, 3
DeMorgan's Theorem, Boolean Algebra, 37-43
Diode switch, logic polarity, 78
Distributive Law, Boolean Algebra, 43-45
Double Negative Law, Boolean Algebra, 33, 34

E

Exclusive OR gate, logic polarity, 85, 86

F

Fractional numbers, conversion of, 13-15
  converting binary fractions to octal fractions, 13
  converting decimal fractions to octal and binary fractions, 14, 15
  converting octal fractions to decimal fractions, 13, 14

G

Glossary of terms, 91-97

H

Hexadecimal system, 4

I

Idempotent Law, Boolean Algebra, 33
Input Inverter, Boolean Algebra, 24-30

L

Law of Absorption, Boolean Algebra, 45-47
Law of Common Identities, Boolean Algebra, 47, 48
Law of Identity, Boolean Algebra, 30
Law of Intersection, Boolean Algebra, 36
Law of Union, Boolean Algebra, 36
Logic circuits, 71-90
  logic polarity, 72-86
    AND gate, 78-81
    diode switch, 78
    exclusive OR gate, 85, 86
    logic one and zero, 74-77
    NAND gate, 84
    NOR gate, 85
    NOT function, 83
    OR gate, 82, 83
    transistor AND gate, 81
  logic switching, 71, 72
  summary of logic circuits, 87-90
Logic one and zero, 74-77
  counting logic pulses, 76
  negative logic, 75
  positive logic, 74

M

Mixed numbers, conversion of, 15, 16

N

NAND gate, Boolean Algebra, 23
NAND gate, logic polarity, 84
Negative logic, logic one and zero, 75
NOR gate, Boolean gate, 23
NOR gate, logic polarity, 85
NOT function, Boolean Algebra, 23
NOT function, logic polarity, 83
Number systems, 1-19
  computers, 1-4
    decimal, binary, octal, and hexadecimal number systems, 3, 4
    historical background, 1
    one-to-one counting, 2
  conversion between hexadecimal and binary, 16, 17
  conversion from decimal to octal and binary, 11, 12
  conversion of fractional numbers, 13-15
  conversion of mixed numbers, 15, 16
  conversion to binary from octal, 10
  conversion to decimal, 7-9
  conversion to octal from binary, 9, 10
  positional notation, 4-7
  summary of number systems, 17-19

O

Octal system, 4
One-to-one counting, 2
OR gate, Boolean Algebra, 21-23
OR gate, logic polarity, 82, 83

P

Positional notation, number systems, 4-7
Positive logic, logic one and zero, 74

T

Transistor AND gate, logic polarity, 81

V

Veitch diagram, Boolean Algebra, 50-62
  summary of the four-variable Veitch diagram, 60
  summary of three-variable Veitch diagrams, 58-60
  thirty-two-square Veitch diagram, 61

★U.S. GOVERNMENT PRINTING OFFICE: 1981—740-086/100

A FINAL QUESTION: What did you think of this course? Of the text material used with the course? Comments and recommendations received from enrollees have been a major source of course improvemen You and your command are urged to submit your constructive criticisms and your recommendations. This tear-out form letter is provided for your convenience. Typewrite if possible, but legible handwriting is acceptable.

Date _____

From: _____
NAME (RANK, RATE, CIVILIAN)

_____

_____ ZIP CODE _____

To: Naval Education and Training Program Development Center Code PD (NEETS)
Pensacola, Florida 32509

Subj: NEETS, NAVEDTRA 172-13-00-79, Module 13

1. The following comments are hereby submitted:

(Fold along dotted line and staple or tape)

(Fold along dotted line and staple or tape)

DEPARTMENT OF THE NAVY
NAVAL EDUCATION AND TRAINING PROGRAM
DEVELOPMENT CENTER PD (NEETS)
PENSACOLA, FLORIDA 32509

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

POSTAGE AND FEES PAID
NAVY DEPARTMENT
DoD-316

U.S. MAIL

NAVAL EDUCATION AND TRAINING PROGRAM DEVELOPMENT CENTER
BUILDING 2435 PD (NEETS)
PENSACOLA, FLORIDA 32509